```
//@ZPayab

//@version=5

indicator(title='DIY Custom Strategy Builder  [ZP] - v1', shorttitle="DIY Custom Strategy Builder  [ZP

ma(_source, _length, _type) =>
    switch _type
        "SMA"  => ta.sma (_source, _length)
        "EMA"  => ta.ema (_source, _length)
        "RMA"  => ta.rma (_source, _length)
        "WMA"  => ta.wma (_source, _length)
        "VWMA" => ta.vwma(_source, _length)


alarm(_osc, _message) =>
    alert(syminfo.ticker + ' ' + _osc + ' : ' + _message + ', price (' + str.tostring(close, format.mi


//Conditional Sampling EMA Function
Cond_EMA(x, cond, n) =>
    var val = array.new_float(0)
    var ema_val = array.new_float(1)
    if cond
        array.push(val, x)
        if array.size(val) > 1
            array.remove(val, 0)
        if na(array.get(ema_val, 0))
            array.fill(ema_val, array.get(val, 0))
        array.set(ema_val, 0, (array.get(val, 0) - array.get(ema_val, 0)) * (2 / (n + 1)) + array.get(
    EMA = array.get(ema_val, 0)
    EMA

//Conditional Sampling SMA Function
Cond_SMA(x, cond, n) =>
    var vals = array.new_float(0)
    if cond
        array.push(vals, x)
        if array.size(vals) > n
            array.remove(vals, 0)
    SMA = array.avg(vals)
    SMA

//Standard Deviation Function
Stdev(x, n) =>
    math.sqrt(Cond_SMA(math.pow(x, 2), 1, n) - math.pow(Cond_SMA(x, 1, n), 2))

//Range Size Function
rng_size(x, scale, qty, n) =>
    ATR = Cond_EMA(ta.tr(true), 1, n)
    AC = Cond_EMA(math.abs(x - x[1]), 1, n)
    SD = Stdev(x, n)

```

```
    rng_size = scale == 'Pips' ? qty * 0.0001 : scale == 'Points' ? qty * syminfo.pointvalue : scale =
    rng_size

//Two Type Range Filter Function
rng_filt(h, l, rng_, n, type, smooth, sn, av_rf, av_n) =>
    rng_smooth = Cond_EMA(rng_, 1, sn)
    r = smooth ? rng_smooth : rng_
    var rfilt = array.new_float(2, (h + l) / 2)
    array.set(rfilt, 1, array.get(rfilt, 0))
    if type == 'Type 1'
        if h - r > array.get(rfilt, 1)
            array.set(rfilt, 0, h - r)
        if l + r < array.get(rfilt, 1)
            array.set(rfilt, 0, l + r)
    if type == 'Type 2'
        if h >= array.get(rfilt, 1) + r
            array.set(rfilt, 0, array.get(rfilt, 1) + math.floor(math.abs(h - array.get(rfilt, 1)) / r
        if l <= array.get(rfilt, 1) - r
            array.set(rfilt, 0, array.get(rfilt, 1) - math.floor(math.abs(l - array.get(rfilt, 1)) / r
    rng_filt1 = array.get(rfilt, 0)
    hi_band1 = rng_filt1 + r
    lo_band1 = rng_filt1 - r
    rng_filt2 = Cond_EMA(rng_filt1, rng_filt1 != rng_filt1[1], av_n)
    hi_band2 = Cond_EMA(hi_band1, rng_filt1 != rng_filt1[1], av_n)
    lo_band2 = Cond_EMA(lo_band1, rng_filt1 != rng_filt1[1], av_n)
    rng_filt = av_rf ? rng_filt2 : rng_filt1
    hi_band = av_rf ? hi_band2 : hi_band1
    lo_band = av_rf ? lo_band2 : lo_band1
    [hi_band, lo_band, rng_filt]

ma_function(source, length, type) =>


    if type == 'RMA'
        ta.rma(source, length)
    else if type == 'SMA'
        ta.sma(source, length)
    else if type == 'EMA'
        ta.ema(source, length)
    else if type == 'WMA'
        ta.wma(source, length)
    else if type == 'HMA'
        if(length<2)
            ta.hma(source,2)
        else
            ta.hma(source, length)
    else
        ta.vwma(source, length)

// Get Table Size
table_size(s) =>
    switch s
        "Auto"   => size.auto
        "Huge"   => size.huge
        "Large"  => size.large
```

```
        "Normal" => size.normal
        "Small"  => size.small
        => size.tiny

setup_group= "▇▇▇▇▇▇▇ Indicator Setup ▇▇▇▇▇▇▇"
signalexpiry = input.int(defval=3,  title='Signal Expiry Candle Count',group=setup_group, inline='expi
alternatesignal = input.bool (true, "Alternate Signal", group=setup_group, inline='alternate')

showsignal = input.bool (true, "Show Long/Short Signal", group=setup_group,inline='showsignal',tooltip

showdashboard = input.bool (true, "Show Dashboard", group=setup_group,inline='dashboard')
string i_tab1Ypos = input.string('bottom', 'Dashboard Position',group=setup_group, inline='dashboard2'
string i_tab1Xpos = input.string('right', '', inline='dashboard2', group=setup_group,options=['left',
in_dashboardtab_size = input.string(title="Dashboard Size    ", defval="Normal",
     options=["Auto",  "Huge",  "Large", "Normal", "Small", "Tiny"],
     group= setup_group , inline= "dashboard3")

////////////////////////////////////////////
///// Signal filters
////////////////////////////////////////////
leadingindicator = input.string(title="Leading Indicator", defval="Range Filter",
 options=["Range Filter", "Rational Quadratic Kernel (RQK)","Supertrend","Half Trend","Ichimoku Cloud"


confirmation_group = "▇▇▇▇▇▇▇ Confirmation Indicators (filter) ▇▇▇▇▇▇▇ "


ema_tooltip = "EMA filter for confirmation.\n\n Validates Long signal if price is above the EMA FILTER
respectema = input.bool (false, "EMA Filter", group=confirmation_group, inline='respectema')
respectemaperiod = input.int(defval=200, minval=1, title='', group=confirmation_group, inline='respect

ema2_tooltip = "Generates Long signal if Fast EMA cross above Slow EMA.\n\n Generates Short signal whe
respect2ma = input.bool (false, "2 EMA Cross : ", group=confirmation_group, inline='2ma')
respect2maperiod_1 = input.int(defval=50,  title='',group=confirmation_group, inline='2ma')
respect2maperiod_2 = input.int(defval=200,  title='',group=confirmation_group, inline='2ma',tooltip=em

ema3_tooltip = "Generates Long signal if first EMA (Fastest) cross above 2nd and 3rd EMA and 2nd EMA c
respect3ma = input.bool (false, "3 EMA Cross : ", group=confirmation_group, inline='3ma',tooltip=ema3_
respect3maperiod_1 = input.int(defval=9,  title='',group=confirmation_group, inline='3ma',tooltip=ema3
respect3maperiod_2 = input.int(defval=21,  title='',group=confirmation_group, inline='3ma',tooltip=ema
respect3maperiod_3 = input.int(defval=55,  title='',group=confirmation_group, inline='3ma',tooltip=ema



respectrf = input.bool (false, "Range Filter", group=confirmation_group, inline='rf')
rftype = input.string(title="", defval="Default", options=["Default","DW"], group=confirmation_group,

respectrqk = input.bool (true, "Rational Quadratic Kernel (RQK)", group=confirmation_group, inline='rq



respectst = input.bool (false, "SuperTrend", group=confirmation_group, inline='st')

respectht = input.bool (false, "Half Trend", group=confirmation_group, inline='ht')
```

```
respectdonchian = input.bool (false, "Donchian Trend Ribbon", group=confirmation_group, inline='donchi

respectroc = input.bool (false, "Rate of Change (ROC)", group=confirmation_group, inline='roc')

respecttsi = input.bool (false, "True Strength Indicator (TSI)", group=confirmation_group, inline='tsi

tsitooltip = "Signal Crossover:\n\n TSI crossover or greater than signal line for long, and TSI crossu
tsitype = input.string(title="", defval="Signal Cross", options=["Signal Cross", "Zero line cross"],to


respecttdfi = input.bool (false, "Trend Direction Force Index (TDFI)", group=confirmation_group, inlir


respectmd = input.bool (false, "McGinley Dynamic", group=confirmation_group, inline='md')

respectdpo = input.bool (false, "Detrended Price Oscillator (DPO)", group=confirmation_group, inline='


respectichi = input.bool (false, "Ichimoku Cloud", group=confirmation_group, inline='ichi')


respectsuperichi = input.bool (false, "SuperIchi", group=confirmation_group, inline='ichi',tooltip="Ic

respecttrendline_breakout = input.bool (false, "Trendline Breakout", group=confirmation_group, inline=

respectrd = input.bool (false, "Range Detector", group=confirmation_group, inline='rd',tooltip="Range

respecthacolt = input.bool (false, "Heiken-Ashi Candlestick Oscillator", group=confirmation_group, inl


respectbx = input.bool (false, "B-Xtrender", group=confirmation_group, inline='bx')
bxtype = input.string(title="", defval="Short and Long term trend", options=["Short and Long term tren


respectbbpt = input.bool (false, "Bull bear Power Trend", group=confirmation_group, inline='bbpt')
bbpttype = input.string(title="", defval="Follow Trend", options=["Follow Trend","Without Trend"], gro


respectvwap = input.bool (false, "VWAP", group=confirmation_group, inline='vwap')

respectbbosc = input.bool (false, "BB Oscillator", group=confirmation_group, inline='bbosc')
bbtype = input.string(title="", defval="Entering Lower/Upper Band", options=["Entering Lower/Upper Ban


respecttm = input.bool (false, "Trend Meter", group=confirmation_group, inline='tm')
tmtype = input.string(title="", defval="3 TM and 2 TB change to same color", options=["3 TM change to


respectce = input.bool (false, "Chandelier Exit", group=confirmation_group, inline='ce')
```

```
respectcci = input.bool (false, "CCI", group=confirmation_group, inline='cci')



respectao = input.bool (false, "Awesome Oscillator", group=confirmation_group, inline='ao')
aotype = input.string(title="", defval="Zero Line Cross", options=["Zero Line Cross","AC Zero Line Cro


respectadx = input.bool (false, "DMI (ADx)", group=confirmation_group, inline='adx')
adxtype = input.string(title="", defval="Adx & +Di -Di", options=["Adx Only","Adx & +Di -Di", "Advance

respectsar = input.bool (false, "Parabolic SAR (PSAR)", group=confirmation_group, inline='sar')


respectwae = input.bool (false, "Waddah Attar Explosion", group=confirmation_group, inline='wae')



vo_tooltip = "Volatility Oscillator: \n\n ====================== \n\n If the spike line is above the
respectvo = input.bool (false, "Volatility Oscillator", group=confirmation_group, inline='vo', tooltip


ci_tooltip = "Choppiness index: \n\n ====================== \n\n If the index is below the defined th
respectci = input.bool (false, "Choppiness Index  ", group=confirmation_group, inline='ci')
ci_limit = input.float(61.8,title=" ", inline='ci',group=confirmation_group, tooltip = ci_tooltip)

respectdv = input.bool (false, "Damiani Volatility (DV)", group=confirmation_group, inline='dv')

dvtype = input.string(title="", defval="Simple", options=["Simple", "Threshold","10p Difference"], gro


stochtooltip="CrossOver:\n-----------------\n\n CrossOver of K and D line at any level. \n\n CrossOve

respectstochastic = input.bool (false, "Stochastic", group=confirmation_group, inline='stoch')
stochtype = input.string(title="", defval="CrossOver", options=["CrossOver", "CrossOver in OB & OS lev
rsi_tooltip = "RSI MA Cross:\n============\n Generate buy signal when RSI cross up RSI MA line and se

respectrsi = input.bool (false, "RSI", group=confirmation_group, inline='rsi')
rsitype = input.string(title="", defval="RSI MA Cross", options=["RSI MA Cross", "RSI Exits OB/OS zone

rsima_tooltip = "RSI MA Direction:\n============\n The buy and sell signal will respect the RSI MA di
respectrsima = input.bool (false, "RSI MA Direction", group=confirmation_group, inline='rsi2',tooltip=

rsilimit_tooltip = "RSI Limit:\n============\n This is to allow you to set limit for the RSI value fo
respectrsilimit = input.bool (false, "RSI Limit : ", group=confirmation_group, inline='rsi3',tooltip=r

rsilimitup = input.int(40, title="Long",inline='rsi3', group=confirmation_group)
rsilimitdown = input.int(60, title="short",inline='rsi3', group=confirmation_group)


rsimalimit_tooltip = "RSI MA Limit:\n============\n This is to allow you to set limit for the RSI MA
respectrsimalimit = input.bool (false, "RSI MA Limit : ", group=confirmation_group, inline='rsi4',tool
```

```
rsimalimitup = input.int(40, title="Long",inline='rsi4', group=confirmation_group)
rsimalimitdown = input.int(60, title="short",inline='rsi4', group=confirmation_group)




macdtooltip="MACD Crossover:\n------------------\n\n CrossOver of MACD and the Signal line. Generates
respectmacd = input.bool (false, "MACD", group=confirmation_group, inline='macd')
macdtype = input.string(title="", defval="MACD Crossover", options=["MACD Crossover", "Zero line cross

respectssl = input.bool (false, "SSL Channel", group=confirmation_group, inline='ssl')

respectstc = input.bool (false, "Schaff Trend Cycle (STC)", group=confirmation_group, inline='stc')
respectchaikin = input.bool (false, "Chaikin Money Flow", group=confirmation_group, inline='chaikin')

respectvol = input.bool (false, "Volume", group=confirmation_group, inline='volume')
volumetype = input.string(title="", defval="volume above MA", options=["volume above MA","Simple", "De

respectwolf = input.bool (false, "Wolfpack Id", group=confirmation_group, inline='wolf')


respectqqe = input.bool (false, "QQE Mod", group=confirmation_group, inline='qqe')
qqetype = input.string(title="", defval="Line", options=["Line", "Bar","Line & Bar"], group=confirmati

respecthull = input.bool (false, "Hull Suite",group=confirmation_group, inline='hull')


respectvi = input.bool (false, "Vortex Indicator",group=confirmation_group, inline='vi')
vitype = input.string(title="", defval="Simple", options=["Simple", "Advance"],group=confirmation_grou
 tooltip = "Simple\n Green Cross Red. \Advance\n     vipcondition := vip > vim and vip > viupper and v
  vimcondition := vip < vim and vim > viupper and vim > vim[1] and vip < vip [1] and vip[1] <= vilower




//////////////////////////////////////////////////////////////////////
// Switch Board
//////////////////////////////////////////////////////////////////////

switchboard_group = "████ Switch Board (Turn On/Off Overlay Indicators) ████"
switch_ema = input.bool (false, "EMA", group=switchboard_group, inline='Switch1')
switch_poi = input.bool (true, "Supply/Demand Zone", group=switchboard_group, inline='Switch1')
switch_sar = input.bool (false, "PSAR", group=switchboard_group, inline='Switch1')
switch_ichi = input.bool (false, "Ichimoku Cloud", group=switchboard_group, inline='Switch2')

switch_ha = input.bool (false, "Heiken-Ashi Candles", group=switchboard_group, inline='Switch2')



switch_rd = input.bool (false, "Range Detector", group=switchboard_group, inline='Switch2')
switch_vwap = input.bool (false, "VWAP", group=switchboard_group, inline='Switch3')
```

```
switch_bb = input.bool (false, "Bollinger Band", group=switchboard_group, inline='Switch3')

switch_supertrend = input.bool (false, "Supertrend", group=switchboard_group, inline='Switch2')
switch_halftrend= input.bool (false, "Half Trend", group=switchboard_group, inline='Switch2')

switch_rangefilter = input.bool (false, "Range Filter", group=switchboard_group, inline='Switch2')


switch_stc = input.bool (false, "STC", group=switchboard_group, inline='Switch3')
switch_pvsra = input.bool (true, "PVSRA", group=switchboard_group, inline='Switch3')
switch_vectorzone = input.bool (false, "Liquidity Zone", group=switchboard_group, inline='Switch3')
switch_fvg = input.bool (false, "Fair Value Gap (FVG)", group=switchboard_group, inline='Switch4')
switch_pivot = input.bool (false, "Pivot Levels", group=switchboard_group, inline='Switch4')
switch_fractal = input.bool (false, "Fractal", group=switchboard_group, inline='Switch4')

bool show_markets = input.bool(true, group=switchboard_group, title='Market Sessions', tooltip='Turn c


////////////////////////////////////////////////////////////////////////
// EMA Selection
////////////////////////////////////////////////////////////////////////

ma_group= "██████████ MAs Line ██████████"
len1bool = input.bool(true,'',group=ma_group,inline='len1')
len1 = input.int(5,  title='MA 1',group=ma_group,inline='len1')
string ma_1_type = input.string(defval='EMA', title='Type', options=['RMA', 'SMA', 'EMA', 'WMA','HMA',
color ma_1_colour = input.color(color.rgb(254, 234, 74, 0), '', inline='len1',group=ma_group)


len2bool = input.bool(true,'',group=ma_group,inline='len2')
len2 = input.int(13, minval=1, title='MA 2',group=ma_group,inline='len2')
string ma_2_type = input.string(defval='EMA', title='Type', options=['RMA', 'SMA', 'EMA', 'WMA','HMA',
color ma_2_colour = input.color(color.rgb(253, 84, 87, 0), '', inline='len2',group=ma_group)


len3bool = input.bool(false,'',group=ma_group,inline='len3')
len3 = input.int(20, minval=1, title='MA 3',group=ma_group,inline='len3')
string ma_3_type = input.string(defval='EMA', title='Type', options=['RMA', 'SMA', 'EMA', 'WMA','HMA',
color ma_3_colour = input.color(color.new(color.aqua, 0), '', inline='len3',group=ma_group)

len4bool = input.bool(true,'',group=ma_group,inline='len4')
len4 = input.int(50, minval=1, title='MA 4',group=ma_group,inline='len4')
string ma_4_type = input.string(defval='EMA', title='Type', options=['RMA', 'SMA', 'EMA', 'WMA','HMA',
color ma_4_colour = input.color(color.new(color.blue, 0), '', inline='len4',group=ma_group)

len5bool = input.bool(true,'',group=ma_group,inline='len5')
len5 = input.int(200, minval=1, title='MA 5',group=ma_group,inline='len5')
string ma_5_type = input.string(defval='EMA', title='Type', options=['RMA', 'SMA', 'EMA', 'WMA','HMA',
color ma_5_colour = input.color(color.new(color.white, 0), '', inline='len5',group=ma_group)



ema1 = request.security(syminfo.tickerid, timeframe.period, ma_function(close, len1, ma_1_type))
ema2 = request.security(syminfo.tickerid, timeframe.period, ma_function(close, len2, ma_2_type))
ema3 = request.security(syminfo.tickerid, timeframe.period, ma_function(close, len3, ma_3_type))
```

```
ema4 = request.security(syminfo.tickerid, timeframe.period, ma_function(close, len4, ma_4_type))
ema5 = request.security(syminfo.tickerid, timeframe.period, ma_function(close, len5, ma_5_type))


plot(len1bool and switch_ema ? ema1:na, color=ma_1_colour, linewidth=2, title='MA 1')
plot(len2bool and switch_ema? ema2:na, color=ma_2_colour, linewidth=2, title='MA 2')
plot(len3bool and switch_ema? ema3:na, color=ma_3_colour, linewidth=2, title='MA 3')
plot(len4bool and switch_ema? ema4:na, color=ma_4_colour, linewidth=2, title='MA 4')
plot(len5bool and switch_ema? ema5:na, color=ma_5_colour, linewidth=2, title='MA 5')




////////////////////////////////////////////////////////////////////////////
////////////// 2EMA cross
////////////////////////////////////////////////////////////////////////////

var float first_2ema = 0
var float second_2ema = 0


if respect2ma or leadingindicator=='2 EMA Cross'
    first_2ema := ta.ema(close, respect2maperiod_1)
    second_2ema := ta.ema(close, respect2maperiod_2)


////////////////////////////////////////////////////////////////////////////
////////////// 3EMA cross
////////////////////////////////////////////////////////////////////////////
var float first_3ema = 0
var float second_3ema = 0
var float third_3ema = 0


if respect3ma  or leadingindicator=='3 EMA Cross'
    first_3ema := ta.ema(close, respect3maperiod_1)
    second_3ema := ta.ema(close, respect3maperiod_2)
    third_3ema := ta.ema(close, respect3maperiod_3)




////////////////////////////////////////////////////////////////////////////
// Pivots
////////////////////////////////////////////////////////////////////////////

pivot_group = '██████████ Pivot Levels ██████████'

AUTO = "Auto"
DAILY = "Daily"
WEEKLY = "Weekly"
MONTHLY = "Monthly"
QUARTERLY = "Quarterly"
YEARLY = "Yearly"
BIYEARLY = "Biyearly"
```

```
TRIYEARLY = "Triyearly"
QUINQUENNIALLY = "Quinquennially"
DECENNIALLY = "Decennially"

TRADITIONAL = "Traditional"
WOODIE = "Woodie"


kind = input.string(title="Type", defval="Traditional", options=[TRADITIONAL], group=pivot_group)
pivot_time_frame = input.string(title="Pivots Timeframe", defval=AUTO, options=[AUTO, DAILY, WEEKLY, M
look_back = input.int(title="Number of Pivots Back", defval=1, minval=1, maxval=5000, group=pivot_grou
is_daily_based = input.bool(title="Use Daily-based Values", defval=true, group=pivot_group, tooltip="W
show_labels = input.bool(title="Show Labels", defval=true, group=pivot_group)
show_prices = input.bool(title="Show Prices", defval=true, group=pivot_group)
position_labels = input.string("Left", "Labels Position", options=["Left", "Right"], group=pivot_group
line_width = input.int(title="Line Width", defval=1, minval=1, maxval=100, group=pivot_group)


var DEF_COLOR = #FB8C00
var arr_time = array.new_int()
var p = array.new_float()
p_color = input.color(DEF_COLOR, "P    ", inline="P", group=pivot_group)
p_show = input.bool(true, "", inline="P", group=pivot_group)
var r1 = array.new_float()
var s1 = array.new_float()
s1_color = input.color(DEF_COLOR, "S1", inline="S1/R1" , group=pivot_group)
s1_show = input.bool(true, "", inline="S1/R1", group=pivot_group)
r1_color = input.color(DEF_COLOR, "              R1", inline="S1/R1", group=pivot_group)
r1_show = input.bool(true, "", inline="S1/R1", group=pivot_group)
var r2 = array.new_float()
var s2 = array.new_float()
s2_color = input.color(DEF_COLOR, "S2", inline="S2/R2", group=pivot_group)
s2_show = input.bool(true, "", inline="S2/R2", group=pivot_group)
r2_color = input.color(DEF_COLOR, "              R2", inline="S2/R2", group=pivot_group)
r2_show = input.bool(true, "", inline="S2/R2", group=pivot_group)
var r3 = array.new_float()
var s3 = array.new_float()
s3_color = input.color(DEF_COLOR, "S3", inline="S3/R3", group=pivot_group)
s3_show = input.bool(true, "", inline="S3/R3", group=pivot_group)
r3_color = input.color(DEF_COLOR, "              R3", inline="S3/R3", group=pivot_group)
r3_show = input.bool(true, "", inline="S3/R3", group=pivot_group)

pivotX_open = float(na)
pivotX_open := nz(pivotX_open[1], open)
pivotX_high = float(na)
pivotX_high := nz(pivotX_high[1], high)
pivotX_low = float(na)
pivotX_low := nz(pivotX_low[1], low)
pivotX_prev_open = float(na)
pivotX_prev_open := nz(pivotX_prev_open[1])
pivotX_prev_high = float(na)
pivotX_prev_high := nz(pivotX_prev_high[1])
pivotX_prev_low = float(na)
pivotX_prev_low := nz(pivotX_prev_low[1])
pivotX_prev_close = float(na)
```

```
pivotX_prev_close := nz(pivotX_prev_close[1])

get_pivot_resolution() =>
    resolution = "M"
    if pivot_time_frame == AUTO
        if timeframe.isintraday
            resolution := timeframe.multiplier <= 15 ? "D" : "W"
        else if timeframe.isweekly or timeframe.ismonthly
            resolution := "12M"
    else if pivot_time_frame == DAILY
        resolution := "D"
    else if pivot_time_frame == WEEKLY
        resolution := "W"
    else if pivot_time_frame == MONTHLY
        resolution := "M"
    else if pivot_time_frame == QUARTERLY
        resolution := "3M"
    else if pivot_time_frame == YEARLY or pivot_time_frame == BIYEARLY or pivot_time_frame == TRIYEARL
        resolution := "12M"
    resolution

var lines = array.new_line()
var labels = array.new_label()

draw_line(i, pivot, col) =>
    if array.size(arr_time) > 1
        array.push(lines, line.new(array.get(arr_time, i), array.get(pivot, i), array.get(arr_time, i

draw_label(i, y, txt, txt_color) =>
    if (show_labels or show_prices) and not na(y)
        display_text = (show_labels ? txt : "") + (show_prices ? str.format(" ({0})", math.round_to_mi
        label_style = position_labels == "Left" ? label.style_label_right : label.style_label_left
        x = position_labels == "Left" ? array.get(arr_time, i) : array.get(arr_time, i + 1)
        array.push(labels, label.new(x = x, y=y, text=display_text, textcolor=txt_color, style=label_s

traditional() =>
    pivotX_Median = (pivotX_prev_high + pivotX_prev_low + pivotX_prev_close) / 3
    array.push(p, pivotX_Median)
    array.push(r1, pivotX_Median * 2 - pivotX_prev_low)
    array.push(s1, pivotX_Median * 2 - pivotX_prev_high)
    array.push(r2, pivotX_Median + 1 * (pivotX_prev_high - pivotX_prev_low))
    array.push(s2, pivotX_Median - 1 * (pivotX_prev_high - pivotX_prev_low))
    array.push(r3, pivotX_Median * 2 + (pivotX_prev_high - 2 * pivotX_prev_low))
    array.push(s3, pivotX_Median * 2 - (2 * pivotX_prev_high - pivotX_prev_low))


calc_pivot() =>
    if kind == TRADITIONAL
        traditional()


resolution = get_pivot_resolution()

SIMPLE_DIVISOR = -1
custom_years_divisor = switch pivot_time_frame
```

```
    BIYEARLY => 2
    TRIYEARLY => 3
    QUINQUENNIALLY => 5
    DECENNIALLY => 10
    => SIMPLE_DIVISOR

calc_high(prev, curr) =>
    if na(prev) or na(curr)
        nz(prev, nz(curr, na))
    else
        math.max(prev, curr)

calc_low(prev, curr) =>
    if not na(prev) and not na(curr)
        math.min(prev, curr)
    else
        nz(prev, nz(curr, na))

calc_OHLC_for_pivot(custom_years_divisor) =>
    if custom_years_divisor == SIMPLE_DIVISOR
        [open, high, low, close, open[1], high[1], low[1], close[1], time[1], time_close]
    else
        var prev_sec_open = float(na)
        var prev_sec_high = float(na)
        var prev_sec_low = float(na)
        var prev_sec_close = float(na)
        var prev_sec_time = int(na)
        var curr_sec_open = float(na)
        var curr_sec_high = float(na)
        var curr_sec_low = float(na)
        var curr_sec_close = float(na)
        if year(time_close) % custom_years_divisor == 0
            curr_sec_open := open
            curr_sec_high := high
            curr_sec_low := low
            curr_sec_close := close
            prev_sec_high := high[1]
            prev_sec_low := low[1]
            prev_sec_close := close[1]
            prev_sec_time := time[1]
            for i = 2 to custom_years_divisor
                prev_sec_open :=  nz(open[i], prev_sec_open)
                prev_sec_high := calc_high(prev_sec_high, high[i])
                prev_sec_low := calc_low(prev_sec_low, low[i])
                prev_sec_time := nz(time[i], prev_sec_time)
        [curr_sec_open, curr_sec_high, curr_sec_low, curr_sec_close, prev_sec_open, prev_sec_high, pre

[sec_open, sec_high, sec_low, sec_close, prev_sec_open, prev_sec_high, prev_sec_low, prev_sec_close, p
sec_open_gaps_on = request.security(syminfo.tickerid, resolution, open, gaps = barmerge.gaps_on, looka

is_change_years = custom_years_divisor > 0 and ta.change(time(resolution)) and year(time_close) % cust

var is_change = false
var uses_current_bar = timeframe.isintraday and kind == WOODIE
var change_time = int(na)
```

```
is_time_change = (ta.change(time(resolution)) and custom_years_divisor == SIMPLE_DIVISOR) or is_change
if is_time_change
    change_time := time

var start_time = time
var was_last_premarket = false
var start_calculate_in_premarket = false

is_last_premarket = barstate.islast and session.ispremarket and time_close > sec_time and not was_last

if is_last_premarket
    was_last_premarket := true
    start_calculate_in_premarket := true
if session.ismarket
    was_last_premarket := false

without_time_change = barstate.islast and array.size(arr_time) == 0
is_can_calc_pivot = (not uses_current_bar and is_time_change and session.ismarket) or (ta.change(sec_c
enough_bars_for_calculate = prev_sec_time >= start_time or is_daily_based

if is_can_calc_pivot and enough_bars_for_calculate and switch_pivot
    if array.size(arr_time) == 0 and is_daily_based
        pivotX_prev_open := prev_sec_open[1]
        pivotX_prev_high := prev_sec_high[1]
        pivotX_prev_low := prev_sec_low[1]
        pivotX_prev_close := prev_sec_close[1]
        pivotX_open := sec_open[1]
        pivotX_high := sec_high[1]
        pivotX_low := sec_low[1]
        array.push(arr_time, start_time)
        calc_pivot()

    if is_daily_based
        if is_last_premarket
            pivotX_prev_open := sec_open
            pivotX_prev_high := sec_high
            pivotX_prev_low := sec_low
            pivotX_prev_close := sec_close
            pivotX_open := open
            pivotX_high := high
            pivotX_low := low
        else
            pivotX_prev_open := prev_sec_open
            pivotX_prev_high := prev_sec_high
            pivotX_prev_low := prev_sec_low
            pivotX_prev_close := prev_sec_close
            pivotX_open := sec_open
            pivotX_high := sec_high
            pivotX_low := sec_low
    else
        pivotX_prev_high := pivotX_high
        pivotX_prev_low := pivotX_low
        pivotX_prev_open := pivotX_open
        pivotX_prev_close := close[1]
        pivotX_open := open
```

```
            pivotX_high := high
            pivotX_low := low

        if barstate.islast and not is_change and array.size(arr_time) > 0 and not without_time_change
            array.set(arr_time, array.size(arr_time) - 1, change_time)
        else if without_time_change
            array.push(arr_time, start_time)
        else
            array.push(arr_time, nz(change_time, time))

        calc_pivot()

        if array.size(arr_time) > look_back
            if array.size(arr_time) > 0
                array.shift(arr_time)
            if array.size(p) > 0 and p_show
                array.shift(p)
            if array.size(r1) > 0 and r1_show
                array.shift(r1)
            if array.size(s1) > 0 and s1_show
                array.shift(s1)
            if array.size(r2) > 0 and r2_show
                array.shift(r2)
            if array.size(s2) > 0 and s2_show
                array.shift(s2)
            if array.size(r3) > 0 and r3_show
                array.shift(r3)
            if array.size(s3) > 0 and s3_show
                array.shift(s3)

        is_change := true
    else if not is_daily_based and switch_pivot
        pivotX_high := math.max(pivotX_high, high)
        pivotX_low := math.min(pivotX_low, low)

    if barstate.islast and not is_daily_based and array.size(arr_time) == 0
        runtime.error("Not enough intraday data to calculate Pivot Points. Lower the Pivots Timeframe or t

    if barstate.islast and array.size(arr_time) > 0 and is_change and switch_pivot
        is_change := false
        if custom_years_divisor > 0
            last_pivot_time = array.get(arr_time, array.size(arr_time) - 1)
            pivot_timeframe = str.tostring(12 * custom_years_divisor) + "M"
            estimate_pivot_time = last_pivot_time + timeframe.in_seconds(pivot_timeframe) * 1000
            array.push(arr_time, estimate_pivot_time)
        else
            array.push(arr_time, time_close(resolution))

        for i = 0 to array.size(lines) - 1
            if array.size(lines) > 0
                line.delete(array.shift(lines))
            if array.size(labels) > 0
                label.delete(array.shift(labels))

        for i = 0 to array.size(arr_time) - 2
```

```
        if array.size(p) > 0 and p_show
            draw_line(i, p, p_color)
            draw_label(i, array.get(p, i), "P", p_color)
        if array.size(r1) > 0 and r1_show
            draw_line(i, r1, r1_color)
            draw_label(i, array.get(r1, i), "R1", r1_color)
        if array.size(s1) > 0 and s1_show
            draw_line(i, s1, s1_color)
            draw_label(i, array.get(s1, i), "S1", s1_color)
        if array.size(r2) > 0 and r2_show
            draw_line(i, r2, r2_color)
            draw_label(i, array.get(r2, i), "R2", r2_color)
        if array.size(s2) > 0 and s2_show
            draw_line(i, s2, s2_color)
            draw_label(i, array.get(s2, i), "S2", s2_color)
        if array.size(r3) > 0 and r3_show
            draw_line(i, r3, r3_color)
            draw_label(i, array.get(r3, i), "R3", r3_color)
        if array.size(s3) > 0 and s3_show
            draw_line(i, s3, s3_color)
            draw_label(i, array.get(s3, i), "S3", s3_color)
```

```
//////////////////////////////////////////////////////////////////////
// William Fractals
//////////////////////////////////////////////////////////////////////
```

```
// Define "n" as the number of periods and keep a minimum value of 2 for error handling.
n = input.int(title="Periods", defval=2, minval=2,group="██████████ Fractal ██████████")
```

```
// UpFractal
bool upflagDownFrontier = true
bool upflagUpFrontier0 = true
bool upflagUpFrontier1 = true
bool upflagUpFrontier2 = true
bool upflagUpFrontier3 = true
bool upflagUpFrontier4 = true

if switch_fractal
    for i = 1 to n
        upflagDownFrontier := upflagDownFrontier and (high[n-i] < high[n])
        upflagUpFrontier0 := upflagUpFrontier0 and (high[n+i] < high[n])
        upflagUpFrontier1 := upflagUpFrontier1 and (high[n+1] <= high[n] and high[n+i + 1] < high[n])
        upflagUpFrontier2 := upflagUpFrontier2 and (high[n+1] <= high[n] and high[n+2] <= high[n] and
        upflagUpFrontier3 := upflagUpFrontier3 and (high[n+1] <= high[n] and high[n+2] <= high[n] and
        upflagUpFrontier4 := upflagUpFrontier4 and (high[n+1] <= high[n] and high[n+2] <= high[n] and
flagUpFrontier = upflagUpFrontier0 or upflagUpFrontier1 or upflagUpFrontier2 or upflagUpFrontier3 or u
```

```
upFractal = (upflagDownFrontier and flagUpFrontier)


// downFractal
bool downflagDownFrontier = true
bool downflagUpFrontier0 = true
bool downflagUpFrontier1 = true
bool downflagUpFrontier2 = true
bool downflagUpFrontier3 = true
bool downflagUpFrontier4 = true

if switch_fractal
    for i = 1 to n
        downflagDownFrontier := downflagDownFrontier and (low[n-i] > low[n])
        downflagUpFrontier0 := downflagUpFrontier0 and (low[n+i] > low[n])
        downflagUpFrontier1 := downflagUpFrontier1 and (low[n+1] >= low[n] and low[n+i + 1] > low[n])
        downflagUpFrontier2 := downflagUpFrontier2 and (low[n+1] >= low[n] and low[n+2] >= low[n] and
        downflagUpFrontier3 := downflagUpFrontier3 and (low[n+1] >= low[n] and low[n+2] >= low[n] and
        downflagUpFrontier4 := downflagUpFrontier4 and (low[n+1] >= low[n] and low[n+2] >= low[n] and
flagDownFrontier = downflagUpFrontier0 or downflagUpFrontier1 or downflagUpFrontier2 or downflagUpFron

downFractal = (downflagDownFrontier and flagDownFrontier)

plotshape(downFractal and switch_fractal ? true : na, style=shape.triangledown, location=location.belo
plotshape(upFractal and switch_fractal ? true : na, style=shape.triangleup,   location=location.abovet
```

```
////////////////////////////////////////////////////////////////////////
// Range Filter
////////////////////////////////////////////////////////////////////////

rf_group= "[redacted] Range Filter [redacted]"

//switch_rangefilter  = input.bool (true, "Show Range Filter Signals", group='=============== Range Fi
showrfline = input.bool (true, "Show RF line", group=rf_group)


src = input.source(defval=close, title='Source', group=rf_group,inline = 'rf')

// Sampling Period
// Settings for 5min chart, BTCUSDC. For Other coin, change the paremeters

per = input.int(defval=100, minval=1, title='Period', group=rf_group,inline = 'rf')

// Range Multiplier

mult = input.float(defval=3.0, minval=0.1, title='Multiplier', group=rf_group,inline = 'rf')
```

```
// Smooth Average Range

smoothrng(x, t, m) =>
    wper = t * 2 - 1
    avrng = ta.ema(math.abs(x - x[1]), t)
    smoothrng = ta.ema(avrng, wper) * m
    smoothrng
smrng = smoothrng(src, per, mult)

// Range Filter

rngfilt(x, r) =>
    rngfilt = x
    rngfilt := x > nz(rngfilt[1]) ? x - r < nz(rngfilt[1]) ? nz(rngfilt[1]) : x - r : x + r > nz(rngfi
    rngfilt
filt = rngfilt(src, smrng)

// Filter Direction

upward = 0.0
upward := filt > filt[1] ? nz(upward[1]) + 1 : filt < filt[1] ? 0 : nz(upward[1])
downward = 0.0
downward := filt < filt[1] ? nz(downward[1]) + 1 : filt > filt[1] ? 0 : nz(downward[1])

// Target Bands

hband = filt + smrng
lband = filt - smrng



////////////////////////////////
////// RF2
////////////////////////////////

//Filter Type
f_type = input.string(defval='Type 1', options=['Type 1', 'Type 2'], title='Filter Type')

//Movement Source
mov_src = input.string(defval='Close', options=['Wicks', 'Close'], title='Movement Source')

//Range Size Inputs
rng_qty = input.float(defval=2.618, minval=0.0000001, title='Range Size')
rng_scale = input.string(defval='Average Change', options=['Points', 'Pips', 'Ticks', '% of Price', 'A

//Range Period
rng_per = input.int(defval=14, minval=1, title='Range Period (for ATR, Average Change, and Standard De

//Range Smoothing Inputs
smooth_range = input(defval=true, title='Smooth Range')
smooth_per = input.int(defval=27, minval=1, title='Smoothing Period')

//Filter Value Averaging Inputs
av_vals = input(defval=false, title='Average Filter Changes')
```

```
av_samples = input.int(defval=2, minval=1, title='Number Of Changes To Average')


//------------------------------------------------------------------------------------------------------------
//Definitions
//------------------------------------------------------------------------------------------------------------

//High And Low Values
h_val = mov_src == 'Wicks' ? high : close
l_val = mov_src == 'Wicks' ? low : close

//Range Filter Values
[h_band, l_band, filt2] = rng_filt(h_val, l_val, rng_size((h_val + l_val) / 2, rng_scale, rng_qty, rng

//Direction Conditions
var fdir2 = 0.0
fdir2 := filt2 > filt2[1] ? 1 : filt2 < filt2[1] ? -1 : fdir2
rfupward = fdir2 == 1 ? 1 : 0
rfdownward= fdir2 == -1 ? 1 : 0


//------------------------------------------------------------------------------------------------------------
//color and switchboard rf
//------------------------------------------------------------------------------------------------------------


filtcolor = upward > 0 ? color.lime : downward > 0  ? color.red : color.orange
// filt2_color = rfupward ? #05ff9b : rfdownward ? #ff0583 : #cccccc

filttype = string(na)
if rftype == "Default"
    filttype := "filt"
else if rftype == "DW"
    filttype := "filt2"


filtplot = plot(switch_rangefilter  and showrfline?filt:na, color=filtcolor, linewidth=2, title='Range
// filtplot2 = plot(switch_rangefilter2 ?filt2:na, color=filt2_color, linewidth=2, title='Range Filter




////////////////////////////////
/// RQK
////////////////////////////////

rqkgroupname = '▇▇▇▇▇▇▇ Rational Quadratic Kernel (RQK) ▇▇▇▇▇▇▇'
rqksrc = input.source(close, 'Source', group=rqkgroupname)
h2 = input.float(8., 'Lookback Window', minval=3., tooltip='The number of bars used for the estimatior
r = input.float(8., 'Relative Weighting', step=0.25, group=rqkgroupname, tooltip='Relative weighting c
x_0 = input.int(25, "Start Regression at Bar", group=rqkgroupname, tooltip='Bar index on which to star
smoothColors = input.bool(false, "Smooth Colors", group=rqkgroupname, tooltip="Uses a crossover based
lag = input.int(2, "Lag", group=rqkgroupname, tooltip="Lag for crossover detection. Lower values resul
size = array.size(array.from(rqksrc)) // size of the data series
```

```
kernel_regression(_rqksrc, _size, _h2) =>
    float _currentWeight = 0.
    float _cumulativeWeight = 0.
    for i = 0 to _size + x_0
        y = _rqksrc[i]
        w = math.pow(1 + (math.pow(i, 2) / ((math.pow(_h2, 2) * 2 * r))), -r)
        _currentWeight += y*w
        _cumulativeWeight += w
    _currentWeight / _cumulativeWeight




var bool rqkuptrend = na
var bool rqkdowntrend = na


if respectrqk or leadingindicator=="Rational Quadratic Kernel (RQK)"
// Estimations
    yhat1 = kernel_regression(rqksrc, size, h2)
    yhat2 = kernel_regression(rqksrc, size, h2-lag)

    // Rates of Change
    bool wasBearish = yhat1[2] > yhat1[1]
    bool wasBullish = yhat1[2] < yhat1[1]
    bool isBearish = yhat1[1] > yhat1
    bool isBullish = yhat1[1] < yhat1
    bool isBearishChange = isBearish and wasBullish
    bool isBullishChange = isBullish and wasBearish

    // Crossovers
    bool isBullishCross = ta.crossover(yhat2, yhat1)
    bool isBearishCross = ta.crossunder(yhat2, yhat1)
    bool isBullishSmooth = yhat2 > yhat1
    bool isBearishSmooth = yhat2 < yhat1


    rqkuptrend := yhat1[1] < yhat1
    rqkdowntrend := yhat1[1] > yhat1




//////////////////////////////
////////// TSI
//////////////////////////////
tsi_group = "████████ True Strength Indicator (TSI) ████████"

tsi_long_length = input(title="Long Length", defval=25,group=tsi_group)
tsi_short_length = input(title="Short Length", defval=13,group=tsi_group)
tsi_signal_length = input(title="Signal Length", defval=13,group=tsi_group)
tsi_price = close
tsi_double_smooth(src, long, short) =>
    fist_smooth = ta.ema(src, long)
```

```
    ta.ema(fist_smooth, short)
tsi_pc = ta.change(tsi_price)
tsi_double_smoothed_pc = tsi_double_smooth(tsi_pc, tsi_long_length, tsi_short_length)
tsi_double_smoothed_abs_pc = tsi_double_smooth(math.abs(tsi_pc), tsi_long_length, tsi_short_length)
tsi_value = 100 * (tsi_double_smoothed_pc / tsi_double_smoothed_abs_pc)

var tsi_long = bool(na)
var tsi_short = bool(na)
tsi_signal = ta.ema(tsi_value, tsi_signal_length)

if tsitype == "Signal Cross"
    tsi_long := tsi_value > tsi_signal
    tsi_short := tsi_value < tsi_signal
else if tsitype == "Zero line cross"
    tsi_long := tsi_value > tsi_signal and tsi_value>0
    tsi_short := tsi_value < tsi_signal and tsi_value<0


////////////////////////////////
///// Super Trend
////////////////////////////////

sp_group = "██████████ SuperTrend ██████████"

Periods = input(title='ATR Period', defval=10, group=sp_group)
stsrc = input(hl2, title='Source', group=sp_group)
Multiplier = input.float(title='ATR Multiplier', step=0.1, defval=3.0, group=sp_group)
changeATR = input(title='Change ATR Calculation Method ?', defval=true, group=sp_group)
showsignals = input(title='Show Buy/Sell Signals ?', defval=true, group=sp_group)
highlighting = input(title='Highlighter On/Off ?', defval=true, group=sp_group)


statr2 = ta.sma(ta.tr, Periods)
statr = changeATR ? ta.atr(Periods) : statr2
stup = stsrc - Multiplier * statr
up1 = nz(stup[1], stup)
stup := close[1] > up1 ? math.max(stup, up1) : stup
dn = stsrc + Multiplier * statr
dn1 = nz(dn[1], dn)
dn := close[1] < dn1 ? math.min(dn, dn1) : dn
sttrend = 1
sttrend := nz(sttrend[1], sttrend)
sttrend := sttrend == -1 and close > dn1 ? 1 : sttrend == 1 and close < up1 ? -1 : sttrend
upPlot = plot(sttrend == 1 and switch_supertrend ? stup : na, title='Up Trend', style=plot.style_lineb
stbuySignal = sttrend == 1 and sttrend[1] == -1
plotshape(stbuySignal and switch_supertrend  ? stup : na, title='UpTrend Begins', location=location.ab
dnPlot = plot(sttrend != 1 and switch_supertrend ? dn : na , title='Down Trend', style=plot.style_line
stsellSignal = sttrend == -1 and sttrend[1] == 1
plotshape(stsellSignal  and switch_supertrend  ? dn : na, title='DownTrend Begins', location=location.


////////////////////////////////////////////
/////////// Half Trend
```

```
//////////////////////////////////////////

ht_group = "████████ HalfTrend ████████"
amplitude = input(title='Amplitude', defval=2,group=ht_group)
channelDeviation = input(title='Channel Deviation', defval=2,group=ht_group)
showArrows = input(title='Show Arrows', defval=true,group=ht_group)
showChannels = input(title='Show Channels', defval=true,group=ht_group)


var bool halftrend_long = na
var bool halftrend_short = na
var float ht = na
var color htColor  = na


if respectht or leadingindicator=="Half Trend" or switch_halftrend
    var int ht_trend = 0
    var int nextTrend = 0
    var float maxLowPrice = nz(low[1], low)
    var float minHighPrice = nz(high[1], high)

    var float ht_up = 0.0
    var float ht_down = 0.0
    float atrHigh = 0.0
    float atrLow = 0.0
    float arrowUp = na
    float arrowDown = na

    ht_atr2 = ta.atr(100) / 2
    ht_dev = channelDeviation * ht_atr2

    highPrice = high[math.abs(ta.highestbars(amplitude))]
    lowPrice = low[math.abs(ta.lowestbars(amplitude))]
    highma = ta.sma(high, amplitude)
    lowma = ta.sma(low, amplitude)

    if nextTrend == 1
        maxLowPrice := math.max(lowPrice, maxLowPrice)

        if highma < maxLowPrice and close < nz(low[1], low)
            ht_trend := 1
            nextTrend := 0
            minHighPrice := highPrice
            minHighPrice
    else
        minHighPrice := math.min(highPrice, minHighPrice)

        if lowma > minHighPrice and close > nz(high[1], high)
            ht_trend := 0
            nextTrend := 1
            maxLowPrice := lowPrice
            maxLowPrice

    if ht_trend == 0
        if not na(ht_trend[1]) and ht_trend[1] != 0
```

```
            ht_up := na(ht_down[1]) ? ht_down : ht_down[1]
            arrowUp := ht_up - ht_atr2
            arrowUp
        else
            ht_up := na(ht_up[1]) ? maxLowPrice : math.max(maxLowPrice, ht_up[1])
            ht_up
        atrHigh := ht_up + ht_dev
        atrLow := ht_up - ht_dev
        atrLow
    else
        if not na(ht_trend[1]) and ht_trend[1] != 1
            ht_down := na(ht_up[1]) ? ht_up : ht_up[1]
            arrowDown := ht_down + ht_atr2
            arrowDown
        else
            ht_down := na(ht_down[1]) ? minHighPrice : math.min(minHighPrice, ht_down[1])
            ht_down
        atrHigh := ht_down + ht_dev
        atrLow := ht_down - ht_dev
        atrLow

    ht := ht_trend == 0 ? ht_up : ht_down

    var color buyColor = color.blue
    var color sellColor = color.red

    htColor := ht_trend == 0 ? buyColor : sellColor



    halftrend_long := ht_trend == 0
    halftrend_short := ht_trend != 0


htPlot = plot(switch_halftrend ? ht:na, title='HalfTrend', linewidth=2, color=htColor)



////////////////////////////////////////////
//////// Trend Line Breakout
////////////////////////////////////////////
//------------------------------------------------------------------------------
//Settings
//------------------------------------------------------------------------------{

tbgroup= "████████ Trendline Breakout ████████"
length_tb = input.int(14, 'Swing Detection Lookback',group=tbgroup)
mult_tb = input.float(1., 'Slope', minval = 0, step = .1,group=tbgroup)
calcMethod = input.string('Atr', 'Slope Calculation Method', options = ['Atr','Stdev','Linreg'],group=
backpaint = input(true, tooltip = 'Backpainting offset displayed elements in the past. Disable backpai



var upos = 0
var dnos = 0
```

```
tb_buysignal = bool (na)
tb_sellsignal = bool (na)

if respecttrendline_breakout or leadingindicator=="Trendline Breakout"

    //------------------------------------------------------------------------------}
    //Calculations
    //------------------------------------------------------------------------------{
    var upper_tb = 0.
    var lower_tb = 0.
    var slope_ph = 0.
    var slope_pl = 0.

    var offset_tb = backpaint ? length_tb : 0

    n_tb = bar_index
    src_tb = close

    ph = ta.pivothigh(length_tb, length_tb)
    pl = ta.pivotlow(length_tb, length_tb)

    //Slope Calculation Method
    slope_tb = switch calcMethod
        'Atr'    => ta.atr(length_tb) / length_tb * mult_tb
        'Stdev'  => ta.stdev(src_tb,length_tb) / length_tb * mult_tb
        'Linreg' => math.abs(ta.sma(src_tb * n_tb, length_tb) - ta.sma(src_tb, length_tb) * ta.sma(n_t

    //Get slopes and calculate trendlines
    slope_ph := ph ? slope_tb : slope_ph
    slope_pl := pl ? slope_tb : slope_pl

    upper_tb := ph ? ph : upper_tb - slope_ph
    lower_tb := pl ? pl : lower_tb + slope_pl


    upos := ph ? 0 : close > upper_tb - slope_ph * length_tb ? 1 : upos
    dnos := pl ? 0 : close < lower_tb + slope_pl * length_tb ? 1 : dnos


    for i = 0 to signalexpiry - 1
        tb_buysignal := upos[0] > upos[i+1]
        tb_sellsignal := dnos[0] > dnos[i+1]









////////////////////////////////////////////
////// Ichimoku
////////////////////////////////////////////

ichigroup = "████████ Ichimoku  ████████"
```

```
conversionPeriods = input.int(9, minval=1, title="Conversion Line Length",group=ichigroup)
basePeriods = input.int(26, minval=1, title="Base Line Length",group=ichigroup)
laggingSpan2Periods = input.int(52, minval=1, title="Leading Span B Length",group=ichigroup)
displacement = input.int(26, minval=1, title="Lagging Span",group=ichigroup)
donchian(len) => math.avg(ta.lowest(len), ta.highest(len))
conversionLine = donchian(conversionPeriods)
baseLine = donchian(basePeriods)
leadLine1 = math.avg(conversionLine, baseLine)
leadLine2 = donchian(laggingSpan2Periods)
ChikouSpan = close[25] + (close - close[25])
lead1 = leadLine1[displacement - 1]
lead2 = leadLine2[displacement - 1]
plot(switch_ichi?conversionLine:na, color=#2962FF, title="Conversion Line",linewidth = 1)
plot(switch_ichi ? baseLine:na, color=#B71C1C, title="Base Line",linewidth = 1)
plot(switch_ichi?close:na, offset = -displacement + 1, color=#43A047, title="Lagging Span")
p1 = plot(switch_ichi?leadLine1:na, offset = displacement - 1, color=#A5D6A7,
     title="Leading Span A")
p2 = plot(switch_ichi?leadLine2:na, offset = displacement - 1, color=#EF9A9A,
      title="Leading Span B")


fill(p1, p2, color = leadLine1 > leadLine2 and switch_ichi ? color.rgb(67, 160, 71, 70) : color.rgb(24



ichi_long = conversionLine > baseLine    and  leadLine1> leadLine2 and close >leadLine1[displacement -
ichi_short = conversionLine < baseLine and leadLine1 < leadLine2 and close < leadLine1[displacement -1



//////////////////////////////
///////// SuperIchi
//////////////////////////////
superichigroup = "██████████ SuperIchi  ██████████"

tenkan_len  = input(9,'Tenkan          ',inline='tenkan',group=superichigroup)
tenkan_mult = input(2.,'',inline='tenkan',group=superichigroup)

kijun_len   = input(26,'Kijun             ',inline='kijun',group=superichigroup)
kijun_mult  = input(4.,'',inline='kijun',group=superichigroup)

spanB_len   = input(52,'Senkou Span B ',inline='span',group=superichigroup)
spanB_mult  = input(6.,'',inline='span',group=superichigroup)

offset      = input(26,'Displacement',group=superichigroup)
//------------------------------------------------------------------------------
avg(src,length,mult)=>
    atr = ta.atr(length)*mult
    up = hl2 + atr
    dn = hl2 - atr
    upper = 0.,lower = 0.
    upper := src[1] < upper[1] ? math.min(up,upper[1]) : up
    lower := src[1] > lower[1] ? math.max(dn,lower[1]) : dn
```

```
    os = 0,max = 0.,min = 0.
    os := src > upper ? 1 : src < lower ? 0 : os[1]
    spt = os == 1 ? lower : upper
    max := ta.cross(src,spt) ? math.max(src,max[1]) : os == 1 ? math.max(src,max[1]) : spt
    min := ta.cross(src,spt) ? math.min(src,min[1]) : os == 0 ? math.min(src,min[1]) : spt
    math.avg(max,min)
//-------------------------------------------------------------------------------
tenkan = avg(close,tenkan_len,tenkan_mult)
kijun = avg(close,kijun_len,kijun_mult)

senkouA = math.avg(kijun,tenkan)
senkouB = avg(close,spanB_len,spanB_mult)
//-------------------------------------------------------------------------------



superichi_long = tenkan > kijun    and  senkouA> senkouB and close >senkouA[displacement -1] and close
superichi_short = tenkan < kijun and senkouA < senkouB and close < senkouA[displacement -1] and close

////////////////////////////////
//////////Donchian Channel Ribbon
////////////////////////////////
donchiangroup = "██████████ Donchian Channel Ribbon ██████████"
dlen = input.int(defval=15, title='Donchian Channel Period', group=donchiangroup)

dchannel(len) =>
    float hh = ta.highest(len)
    float ll = ta.lowest(len)

    int trend = 0
    trend := close > hh[1] ? 1 : close < ll[1] ? -1 : nz(trend[1])
    trend

dchannelalt(len, maintrend) =>
    float hh = ta.highest(len)
    float ll = ta.lowest(len)

    int trend = 0
    trend := close > hh[1] ? 1 : close < ll[1] ? -1 : nz(trend[1])
    maintrend == 1 ? trend == 1 ? #00FF00ff : #00FF009f : maintrend == -1 ? trend == -1 ? #FF0000ff :

maintrend = dchannel(dlen)

donchian_long = maintrend == 1 ? true:na
donchian_short = maintrend == -1 ? true:na


////////////////////////////////
// DMI code
////////////////////////////////

dmi_group = "████████ Directional Movement Index (DMI) ████████"
adxlen = input(5, title="ADX", group=dmi_group, inline='dmi')
keyLevel = input(20, title='ADX limit',group=dmi_group, inline='dmi')
dilen = input.int(10, title="DI Length",group=dmi_group, inline='dmi')
```

```
lensig = adxlen
upp = ta.change(high)
downn = ta.change(low)
plusDM = na(upp) ? na : upp > downn and upp > 0 ? upp : 0
minusDM = na(downn) ? na : downn > upp and downn > 0 ? downn : 0
trur = ta.rma(ta.tr, dilen)
plus = fixnan(100 * ta.rma(plusDM, dilen) / trur)
minus = fixnan(100 * ta.rma(minusDM, dilen) / trur)
sum = plus + minus
adxx = 100 * ta.rma(math.abs(plus - minus) / (sum == 0 ? 1 : sum), lensig)
dirmov(dilen) =>
    up = ta.change(high)
    down = -ta.change(low)
    truerange = ta.rma(ta.tr, dilen)
    plus = fixnan(100 * ta.rma(up > down and up > 0 ? up : 0, dilen) / truerange)
    minus = fixnan(100 * ta.rma(down > up and down > 0 ? down : 0, dilen) / truerange)
    [plus, minus]

adx(dilen, adxlen) =>
    [plus, minus] = dirmov(dilen)
    sum = plus + minus
    adx = 100 * ta.rma(math.abs(plus - minus) / (sum == 0 ? 1 : sum), adxlen)
    [adx, plus, minus]

[adx, diplus, diminus] = adx(dilen, adxlen)

////////////////////////////////////////////
//Parabolic SAR
////////////////////////////////////////////
//showsar = input.bool (false, "Show SAR", group='-------------------------Parabolic SAR--------------
psar_group = "██████████ Parabolic SAR (PSAR) ██████████"
start = input.float(0.02, group=psar_group, inline ='sar')
increment = input.float(0.02, group=psar_group, inline ='sar')
maximum = input.float(0.2, 'Max Value', group=psar_group, inline ='sar')
out = ta.sar(start, increment, maximum)
sarcolor = if (out>close)
    color.red
else
    color.green

if (switch_sar )
    color.red
else
    color.green
plot(switch_sar  ? out : na, 'ParabolicSAR', style=plot.style_cross, color=sarcolor)



////////////////////////////////////////////////
///////// TDFI
////////////////////////////////////////////////
rdfi_group = "█████ Trend Direction Force Index (TDFI) ████ "
```

```
lookback = input(13, title='Lookback',group=rdfi_group)
mmaLength = input(13, title='MMA Length',group=rdfi_group)
mmaMode = input.string(title='MMA Mode', defval='ema', options=['ema', 'wma', 'swma', 'vwma', 'hull',
smmaLength = input(13,group=rdfi_group)
smmaMode = input.string(title='SMMA Mode', defval='ema', options=['ema', 'wma', 'swma', 'vwma', 'hull'
nLength = input(3, title='N Length',group=rdfi_group)
filterHigh = input(0.05, title='Filter High',group=rdfi_group)
filterLow = input(-0.05, title='Filter Low',group=rdfi_group)
price = input(close, 'Period',group=rdfi_group)

tema(src, len) =>
    ema1_tdfi = ta.ema(src, len)
    ema2_tdfi = ta.ema(ema1_tdfi, len)
    ema3_tdfi = ta.ema(ema2_tdfi, len)
    3 * ema1_tdfi - 3 * ema2_tdfi + ema3_tdfi

tdfi_ma(mode, src, len) =>
    mode == 'ema' ? ta.ema(src, len) : mode == 'wma' ? ta.wma(src, len) : mode == 'swma' ? ta.swma(src

tdfi() =>
    mma = tdfi_ma(mmaMode, price * 1000, mmaLength)
    smma = tdfi_ma(smmaMode, mma, smmaLength)
    impetmma = mma - mma[1]
    impetsmma = smma - smma[1]
    divma = math.abs(mma - smma)
    averimpet = (impetmma + impetsmma) / 2
    tdf = math.pow(divma, 1) * math.pow(averimpet, nLength)
    tdf / ta.highest(math.abs(tdf), lookback * nLength)

signal_tdfi = tdfi()

tdfi_long = signal_tdfi > filterHigh
tdfi_short = signal_tdfi < filterLow







////////////////////////////////////////////////
//////// McGinley Dynamic
////////////////////////////////////////////////
md_group = "████████████████ McGinley Dynamic ████████████████"


md_length = input.int(14, minval=1,group=md_group)
md_source = close
mg = 0.0
mg := na(mg[1]) ? ta.ema(md_source, md_length) : mg[1] + (md_source - mg[1]) / (md_length * math.pow(m


md_long = close > mg
md_short = close<mg
```

```
////////////////////////////////////////////////
//////// CCI
////////////////////////////////////////////////
cci_group = "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ CCI ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
ccilength = input.int(20,title="CCI Length", minval=1,inline="cci", group=cci_group)
ccisrc = input(hlc3, title="Source",inline="cci", group=cci_group)
cciupperband = input.int(100,title="Upper Band",inline="cci2", group=cci_group)
ccilowerband = input.int(-100,title="Lower Band",inline="cci2", group=cci_group)

ma = ta.sma(ccisrc, ccilength)
cci = (ccisrc - ma) / (0.015 * ta.dev(ccisrc, ccilength))



typeMA = input.string(title = "Method", defval = "SMA", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "V
smoothingLength = input.int(title = "Length", defval = 5, minval = 1, maxval = 100, group="Smoothing",

smoothingLine = ma(cci, smoothingLength, typeMA)


ccilong = cci > cciupperband
ccishort = cci < ccilowerband




////////////////////////////////////////////////
/////// B-Xtrender
////////////////////////////////////////////////

bxgroup= "▒▒▒▒▒▒▒▒▒▒ B-Xtrender ▒▒▒▒▒▒▒▒▒▒"
short_l1 = input(5, title='Short - L1',group=bxgroup)
short_l2 = input(20, title='Short - L2',group=bxgroup)
short_l3 = input(15, title='Short - L3',group=bxgroup)

long_l1 = input(5, title='Long - L1',group=bxgroup)
long_l2 = input(10, title='Long - L2',group=bxgroup)

shortTermXtrender = ta.rsi(ta.ema(close, short_l1) - ta.ema(close, short_l2), short_l3) - 50
longTermXtrender = ta.rsi(ta.ema(close, long_l1), long_l2) - 50


t3(src, len) =>
    xe1_1 = ta.ema(src, len)
    xe2_1 = ta.ema(xe1_1, len)
    xe3_1 = ta.ema(xe2_1, len)
    xe4_1 = ta.ema(xe3_1, len)
    xe5_1 = ta.ema(xe4_1, len)
    xe6_1 = ta.ema(xe5_1, len)
    b_1 = 0.7
    c1_1 = -b_1 * b_1 * b_1
    c2_1 = 3 * b_1 * b_1 + 3 * b_1 * b_1 * b_1
```

```
    c3_1 = -6 * b_1 * b_1 - 3 * b_1 - 3 * b_1 * b_1 * b_1
    c4_1 = 1 + 3 * b_1 + b_1 * b_1 * b_1 + 3 * b_1 * b_1
    nT3Average_1 = c1_1 * xe6_1 + c2_1 * xe5_1 + c3_1 * xe4_1 + c4_1 * xe3_1
    nT3Average_1

maShortTermXtrender = t3(shortTermXtrender, 5)


bx_long = bool(na)
bx_short = bool(na)

if bxtype == "Short Term trend"
    bx_long := maShortTermXtrender > maShortTermXtrender[1]
    bx_short := maShortTermXtrender < maShortTermXtrender[1]
else if bxtype == "Short and Long term trend"
    bx_long := maShortTermXtrender > maShortTermXtrender[1] and (longTermXtrender > 0 and longTermXtre
    bx_short := maShortTermXtrender < maShortTermXtrender[1]   and (longTermXtrender < 0 and longTermX



////////////////////////////////////////////
////// Bull Bear Power Trend (BBPT)
////////////////////////////////////////////


BullTrend_hist = 0.0
BearTrend_hist = 0.0

BullTrend = (close - ta.lowest(low, 50)) / ta.atr(5)
BearTrend = (ta.highest(high, 50) - close) / ta.atr(5)
BearTrend2 = -1 * BearTrend



Trend = BullTrend - BearTrend

if BullTrend < 2
    BullTrend_hist := BullTrend - 2
    BullTrend_hist

if BearTrend2 > -2
    BearTrend_hist := BearTrend2 + 2
    BearTrend_hist


bbpt_long = bool(na)
bbpt_short = bool(na)
if bbpttype =="Follow Trend"
    bbpt_long := BearTrend_hist > 0 and Trend>=2
    bbpt_short := BullTrend_hist < 0  and Trend<=-2
else if bbpttype == "Without Trend"
    bbpt_long := BearTrend_hist > 0
    bbpt_short := BullTrend_hist < 0
```

```
//////////////////////////////////////////////
//////////// VWAP
//////////////////////////////////////////////

vwap_group = "████████ VWAP Settings ████████"
hideonDWM = input(false, title="Hide VWAP on 1D or Above", group=vwap_group)
var anchor = input.string(defval = "Session", title="Anchor Period",
 options=["Session", "Week", "Month", "Quarter", "Year", "Decade", "Century", "Earnings", "Dividends",
srcvwap = input(title = "Source", defval = hlc3, group=vwap_group)
offsetvwap = input(0, title="Offset", group=vwap_group)

showBand_1 = input(true, title="", group="Standard Deviation Bands Settings", inline="band_1")
stdevMult_1 = input(1.0, title="Bands Multiplier #1", group="Standard Deviation Bands Settings", inlir
showBand_2 = input(false, title="", group="Standard Deviation Bands Settings", inline="band_2")
stdevMult_2 = input(2.0, title="Bands Multiplier #2", group="Standard Deviation Bands Settings", inlir
showBand_3 = input(false, title="", group="Standard Deviation Bands Settings", inline="band_3")
stdevMult_3 = input(3.0, title="Bands Multiplier #3", group="Standard Deviation Bands Settings", inlir

if barstate.islast and ta.cum(volume) == 0
    runtime.error("No volume is provided by the data vendor.")

new_earnings = request.earnings(syminfo.tickerid, earnings.actual, barmerge.gaps_on, barmerge.lookahea
new_dividends = request.dividends(syminfo.tickerid, dividends.gross, barmerge.gaps_on, barmerge.lookah
new_split = request.splits(syminfo.tickerid, splits.denominator, barmerge.gaps_on, barmerge.lookahead_

isNewPeriod = switch anchor
    "Earnings"  => not na(new_earnings)
    "Dividends" => not na(new_dividends)
    "Splits"    => not na(new_split)
    "Session"   => timeframe.change("D")
    "Week"      => timeframe.change("W")
    "Month"     => timeframe.change("M")
    "Quarter"   => timeframe.change("3M")
    "Year"      => timeframe.change("12M")
    "Decade"    => timeframe.change("12M") and year % 10 == 0
    "Century"   => timeframe.change("12M") and year % 100 == 0
    => false

isEsdAnchor = anchor == "Earnings" or anchor == "Dividends" or anchor == "Splits"
if na(srcvwap[1]) and not isEsdAnchor
    isNewPeriod := true

float vwapValue = na
float upperBandValue1 = na
float lowerBandValue1 = na
float upperBandValue2 = na
float lowerBandValue2 = na
float upperBandValue3 = na
float lowerBandValue3 = na

if not (hideonDWM and timeframe.isdwm)
    [_vwap, _stdevUpper, _] = ta.vwap(srcvwap, isNewPeriod, 1)
    vwapValue := _vwap
    stdevAbs = _stdevUpper - _vwap
    upperBandValue1 := _vwap + stdevAbs * stdevMult_1
```

```
    lowerBandValue1 := _vwap - stdevAbs * stdevMult_1
    upperBandValue2 := _vwap + stdevAbs * stdevMult_2
    lowerBandValue2 := _vwap - stdevAbs * stdevMult_2
    upperBandValue3 := _vwap + stdevAbs * stdevMult_3
    lowerBandValue3 := _vwap - stdevAbs * stdevMult_3

plot(switch_vwap? vwapValue:na, title="VWAP", color=#2962FF, offset=offsetvwap)


long_vwap = close>vwapValue
short_vwap = close < vwapValue




////////////////////////////////////////////
////// Chandelier Exit
////////////////////////////////////////////


ChandelierE = "██████████ Chandelier Exit ██████████"

ce_length = input(title='ATR Period', defval=22,group=ChandelierE)
ce_mult = input.float(title='ATR Multiplier', step=0.1, defval=3.0,group=ChandelierE)
showLabels = input(title='Show Buy/Sell Labels ?', defval=true,group=ChandelierE)
useClose = input(title='Use Close Price for Extremums ?', defval=true,group=ChandelierE)
highlightState = input(title='Highlight State ?', defval=true,group=ChandelierE)

ce_atr = ce_mult * ta.atr(ce_length)

longStop = (useClose ? ta.highest(close, ce_length) : ta.highest(ce_length)) - ce_atr
longStopPrev = nz(longStop[1], longStop)
longStop := close[1] > longStopPrev ? math.max(longStop, longStopPrev) : longStop

shortStop = (useClose ? ta.lowest(close, ce_length) : ta.lowest(ce_length)) + ce_atr
shortStopPrev = nz(shortStop[1], shortStop)
shortStop := close[1] < shortStopPrev ? math.min(shortStop, shortStopPrev) : shortStop

var int dir = 1
dir := close > shortStopPrev ? 1 : close < longStopPrev ? -1 : dir


ce_long = dir == 1
ce_short = dir == -1




////////////////////////////////////////////
////// ROC
////////////////////////////////////////////
```

```
roc_group = "████████ Rate of Change (ROC) ████████"


roc_length = input.int(9, minval=1,group=roc_group)
roc_source = input(close, "Source",group=roc_group)
roc = 100 * (roc_source - roc_source[roc_length])/roc_source[roc_length]
roc_long= roc>0
roc_short = roc<0



////////////////////////////////////////////
////// SSL Channel
////////////////////////////////////////////


group_ssl = "████████ SSL Channel ████████"
SSLperiod = input(title='Period', defval=10, group=group_ssl)
SSLlen = input(title='Period', defval=10, group=group_ssl)
smaHigh = ta.sma(high, SSLlen)
smaLow = ta.sma(low, SSLlen)
Hlv = int(na)
Hlv := close > smaHigh ? 1 : close < smaLow ? -1 : Hlv[1]
sslDown = Hlv < 0 ? smaHigh : smaLow
sslUp = Hlv < 0 ? smaLow : smaHigh

ssl_long = sslUp>sslDown
ssl_short = sslUp<sslDown
////////////////////////////////////////////
////// Chaikin Money Flow
////////////////////////////////////////////

group_chaikin = "████████ Chaiken Money Flow ████████"
chaiking_length = input.int(20, minval=1, group = group_chaikin )
ad = close == high and close == low or high == low ? 0 : (2 * close - low - high) / (high - low) * vol
mf = math.sum(ad, chaiking_length) / math.sum(volume, chaiking_length)

chaikin_long = mf>0
chaikin_short = mf<0

////////////////////////////////////////////
////// Vortex INdex
////////////////////////////////////////////
vortex_group = "████████ Vortex Index ████████"

period_ = input.int(14, title="Length", minval=2, group=vortex_group, inline = 'vi')
viupper = input.float(1.1, title="Upper band",group=vortex_group, inline = 'vi')
vilower = input.float(0.9, title="Lower Band",  group=vortex_group, inline = 'vi')

VMP = math.sum( math.abs( high - low[1]), period_ )
VMM = math.sum( math.abs( low - high[1]), period_ )
STR = math.sum( ta.atr(1), period_ )
vip = VMP / STR
vim = VMM / STR
```

```
////////////////////////////////////////////////
////////Waddar Atar explosion (WAR)
////////////////////////////////////////////////
group_wae = "████████ Waddah Attar Explosion ████████"


wae_sensitivity = input(150, title="Sensitivity",group=group_wae)
wae_fastLength=input(20, title="FastEMA Length",group=group_wae)
wae_slowLength=input(40, title="SlowEMA Length",group=group_wae)
channelLength=input(20, title="BB Channel Length",group=group_wae)
wae_mult=input(2.0, title="BB Stdev Multiplier",group=group_wae)

deadzone = nz(ta.rma(ta.tr(true),100)) * 3.7

calc_macd(source, wae_fastLength, wae_slowLength) =>
    fastMA = ta.ema(source, wae_fastLength)
    slowMA = ta.ema(source, wae_slowLength)
    fastMA - slowMA

calc_BBUpper(source, length, wae_mult) =>
    basis = ta.sma(source, length)
    dev = wae_mult * ta.stdev(source, length)
    basis + dev

calc_BBLower(source, length, wae_mult) =>
    basis = ta.sma(source, length)
    dev = wae_mult * ta.stdev(source, length)
    basis - dev

t1 = (calc_macd(close, wae_fastLength, wae_slowLength) - calc_macd(close[1], wae_fastLength, wae_slowl

e1 = (calc_BBUpper(close, channelLength, wae_mult) - calc_BBLower(close, channelLength, wae_mult))

trendUp = (t1 >= 0) ? t1 : 0
trendDown = (t1 < 0) ? (-1*t1) : 0


wae_long = trendUp and trendUp >e1 and e1 > deadzone and trendUp>deadzone
wae_short = trendDown  and trendDown >e1 and e1 > deadzone and trendDown>deadzone




////////////////////////////////////////////
////////// Range Detector
////////////////////////////////////////////

rdgroup= "████████ Range Detector ████████"


//-------------------------------------------------------------------------------
//Settings
//-------------------------------------------------------------------------------{
```

```
rd_length = input.int(20, 'Minimum Range Length', minval = 2,group=rdgroup)
rd_mult   = input.float(1., 'Range Width', minval = 0, step = 0.1,group=rdgroup)
rd_atrLen = input.int(500, 'ATR Length', minval = 1,group=rdgroup)
//Style
rd_upCss = input(#089981, 'Broken Upward', group = 'Style',group=rdgroup)
rd_dnCss = input(#f23645, 'Broken Downward', group = 'Style',group=rdgroup)
unbrokenCss = input(#2157f3, 'Unbroken', group = 'Style',group=rdgroup)

//---------------------------------------------------------------------------------}
//Detect and highlight ranges
//---------------------------------------------------------------------------------{
//Ranges drawings
var box bx = na
var line lvl = na

//Extensions
var float rd_max = na
var float rd_min = na

var rd_os = 0
color detect_css = na

rd_n = bar_index
rd_atr = ta.atr(rd_atrLen) * rd_mult
rd_ma = ta.sma(close, rd_length)

count = 0
for i = 0 to rd_length-1
    count += math.abs(close[i] - rd_ma) > rd_atr ? 1 : 0

// if  switch_rd
if count == 0 and count[1] != count
    //Test for overlap and change coordinates
    if rd_n[rd_length] <= bx.get_right()
        rd_max := math.max(rd_ma + rd_atr, bx.get_top())
        rd_min := math.min(rd_ma - rd_atr, bx.get_bottom())

        if switch_rd
            //Box new coordinates
            bx.set_top(rd_max)
            bx.set_rightbottom(rd_n, rd_min)
            bx.set_bgcolor(color.new(unbrokenCss, 80))

            //Line new coordinates
            avg = math.avg(rd_max, rd_min)
            lvl.set_y1(avg)
            lvl.set_xy2(rd_n, avg)
            lvl.set_color(unbrokenCss)
    else
        rd_max := rd_ma + rd_atr
        rd_min := rd_ma - rd_atr

        //Set new box and level
        if switch_rd
            bx := box.new(rd_n[rd_length], rd_ma + rd_atr, rd_n, rd_ma - rd_atr, na, bgcolor = color.r
```

```
            lvl := line.new(rd_n[rd_length], rd_ma, rd_n, rd_ma, color = unbrokenCss, style = line.sty

            detect_css := color.new(color.gray, 80)
            rd_os := 0

    else if count == 0
        bx.set_right(rd_n)
        lvl.set_x2(rd_n)

    //Set color
    if close > bx.get_top()
        bx.set_bgcolor(color.new(rd_upCss, 80))
        lvl.set_color(rd_upCss)
        rd_os := 1
    else if close < bx.get_bottom()
        bx.set_bgcolor(color.new(rd_dnCss, 80))
        lvl.set_color(rd_dnCss)
        rd_os := -1

    //-------------------------------------------------------------------------------}
    //Plots
    //-------------------------------------------------------------------------------{
    //Range detection bgcolor
    bgcolor(detect_css)

    plot(switch_rd? rd_max:na, 'Range Top'
      , rd_max != rd_max[1] ? na : rd_os == 0 ? unbrokenCss : rd_os == 1 ? rd_upCss : rd_dnCss)

    plot(switch_rd ? rd_min:na, 'Range Bottom'
      , rd_min != rd_min[1] ? na : rd_os == 0 ? unbrokenCss : rd_os == 1 ? rd_upCss : rd_dnCss)

    //-------------------------------------------------------------------------------}




    // for leading indicator
    rd_long = close > rd_max
    rd_short = close < rd_min


    //for confirmation indicator
    rd_signal = close > rd_max or close <rd_min




    ////////////////////////////////////////////////
    ////////Volatility Oscillator
    ////////////////////////////////////////////////
    group_vo = "██████████ Volatility Oscillator ██████████"
    volength = input(100, group = group_vo)
    spike = close - open
```

```
vox = ta.stdev(spike,volength)
voy = ta.stdev(spike,volength) * -1

vo_long = spike > vox
vo_short = spike < voy


////////////////////////////////////////////////
////////Detrended Price Oscillator (DPO)
////////////////////////////////////////////////
group_dpo = "█████████ Detrended Price Oscillator (DPO) █████████"

dpo_period_ = input.int(10, title="Length", minval=1,group=group_dpo)
isCentered = input(false, title="Centered",group=group_dpo)
barsback = dpo_period_/2 + 1
dpo_ma = ta.sma(close, dpo_period_)
dpo = isCentered ? close[barsback] - dpo_ma : close - dpo_ma[barsback]


dpo_long = dpo > 0
dpo_short = dpo<0







////////////////////////////////////////////////
////////HACOLT
////////////////////////////////////////////////
hacolt_group = "███████ Heiken-Ashi Candlestick Oscillator ███████"

hacolt_length = input(defval=55, title="TEMA Period",group=hacolt_group)
hacolt_emaLength = input(defval=60, title="EMA Period",group=hacolt_group)
hacolt_candleSizeFactor = input(defval=1.1, title="Candle size factor",group=hacolt_group)

calc_tema(src, length) =>
    hacolt_ema1 = ta.ema(src, length)
    hacolt_ema2 = ta.ema(hacolt_ema1, length)
    hacolt_ema3 = ta.ema(hacolt_ema2, length)
    3 * (hacolt_ema1 - hacolt_ema2) + hacolt_ema3

var bool hacolt_long = na
var bool hacolt_short = na

if respecthacolt or leadingindicator == "Heiken-Ashi Candlestick Oscillator"
    var float hacolt_haOpen = na
    hacolt_haOpen := nz(hacolt_haOpen[1] + hl2) / 2
    hacolt_haClose = (hacolt_haOpen + math.max(high, hacolt_haOpen) + math.min(low, hacolt_haOpen) + h
    hacolt_thaClose = calc_tema(hacolt_haClose, hacolt_length)
    hacolt_thl2 = calc_tema(hl2, hacolt_length)
    hacolt_haCloseSmooth = 2 * hacolt_thaClose - calc_tema(hacolt_thaClose, hacolt_length)
    hacolt_hl2Smooth = 2 * hacolt_thl2 - calc_tema(hacolt_thl2, hacolt_length)
    hacolt_shortCandle = math.abs(close - open) < ((high - low) * hacolt_candleSizeFactor)
    hacolt_keepn1 = ((hacolt_haClose >= hacolt_haOpen) and (hacolt_haClose[1] >= hacolt_haOpen[1])) or
```

```
    hacolt_keepall1 = hacolt_keepn1 or (hacolt_keepn1[1] and (close >= open) or (close >= close[1]))
    hacolt_keep13 = hacolt_shortCandle and (high >= low[1])
    hacolt_utr = hacolt_keepall1 or (hacolt_keepall1[1] and hacolt_keep13)
    hacolt_keepn2 = (hacolt_haClose < hacolt_haOpen) and (hacolt_haClose[1] < hacolt_haOpen[1]) or (ha
    hacolt_keep23 = hacolt_shortCandle and (low <= high[1])
    hacolt_keepall2 = hacolt_keepn2 or (hacolt_keepn2[1] and (close < open) or (close < close[1]))
    hacolt_dtr = hacolt_keepall2 or (hacolt_keepall2[1] and hacolt_keep23)
    hacolt_upw = hacolt_dtr == 0 and hacolt_dtr[1] and hacolt_utr
    hacolt_dnw = hacolt_utr == 0 and hacolt_utr[1] and hacolt_dtr
    var bool hacolt_upwWithOffset = na
    hacolt_upwWithOffset := hacolt_upw != hacolt_dnw ? hacolt_upw : nz(hacolt_upwWithOffset[1])

    hacolt_buySig = hacolt_upw or (not hacolt_dnw and (na(hacolt_upwWithOffset) ? false : hacolt_upwWi
    hacolt_ltSellSig = close < ta.ema(close, hacolt_emaLength)
    var bool hacolt_neutralSig = na
    hacolt_neutralSig := hacolt_buySig or (hacolt_ltSellSig ? false : nz(hacolt_neutralSig[1]))
    hacolt = hacolt_buySig ? 1 : hacolt_neutralSig ? 0 : -1

    hacolt_long := hacolt > 0
    hacolt_short := hacolt < 0
```

```
////////////////////////////////////////////////
////////Choppiness Index
////////////////////////////////////////////////
group_ci = "██████████ Choppiness Index ██████████"


ci_length = input.int(14, minval=1,group=group_ci)


ci_index = 100 * math.log10(math.sum(ta.atr(1), ci_length) / (ta.highest(ci_length) - ta.lowest(ci_ler

ci_filter = ci_index<ci_limit
```

```
////////////////////////////////////////////////
////////Damiani Volatmeter
////////////////////////////////////////////////

dv_group = "█████████ Damiani Volatmeter █████████"
int vis_atr = input.int(13,title="Vis ATR", group=dv_group, inline = '1')
int vis_std = input.int(20,title="Vis STD", group=dv_group, inline = '1')
int sed_atr = input.int(40,title="Sed ATR", group=dv_group, inline = '1')
int sed_std = input.int(100,title="SEd STD", group=dv_group, inline = '1')
float threshold_level = input.float(1.4,title="Threshold", group=dv_group, inline = '1')
bool lag_supressor = input.bool(true,title="Lag Supressor", group=dv_group, inline = '1')
lag_s_K = 0.5


vol = 0.0
s1_pivot=nz(vol[1], 0)
s3_pivot=nz(vol[3], 0)

vol := lag_supressor ? ta.atr(vis_atr) / ta.atr(sed_atr) + lag_s_K*(s1_pivot-s3_pivot) : ta.atr(vis_at
anti_thres = ta.stdev(close, vis_std) / ta.stdev(close, sed_std)
t = threshold_level - anti_thres
vol_m = vol > t ? -1 : 0.03




////////////////////////////////////////////
/////// MACD
////////////////////////////////////////////
macd_group = "█████████ MACD █████████"
fast_length = input(title="Fast Length", defval=12,group=macd_group)
slow_length = input(title="Slow Length", defval=26,group=macd_group)
macdsrc = input(title="Source", defval=close,group=macd_group)
signal_length = input.int(title="Signal Smoothing",  minval = 1, maxval = 50, defval = 9,group=macd_gr
sma_source = input.string(title="Oscillator MA Type",  defval="EMA", options=["SMA", "EMA"],group=macc
sma_signal = input.string(title="Signal Line MA Type", defval="EMA", options=["SMA", "EMA"],group=macc

fast_ma = sma_source == "SMA" ? ta.sma(macdsrc, fast_length) : ta.ema(macdsrc, fast_length)
slow_ma = sma_source == "SMA" ? ta.sma(macdsrc, slow_length) : ta.ema(macdsrc, slow_length)
macdd = fast_ma - slow_ma
signal = sma_signal == "SMA" ? ta.sma(macdd, signal_length) : ta.ema(macdd, signal_length)
hist = macdd - signal




////////////////////////////////////////////
///// Awesome Oscillator
////////////////////////////////////////////

ao_group = "█████████ Awesome Oscillator █████████"
nLengthSlow = input(34, title="Length Slow",group=ao_group)
nLengthFast = input(5,  title="Length Fast",group=ao_group)
```

```
reverse = input(false, title="Trade reverse",group=ao_group)
xSMA1_hl2 = ta.sma(hl2, nLengthFast)
xSMA2_hl2 = ta.sma(hl2, nLengthSlow)
xSMA1_SMA2 = xSMA1_hl2 - xSMA2_hl2
xSMA_hl2 = ta.sma(xSMA1_SMA2, nLengthFast)
nRes =  xSMA1_SMA2 - xSMA_hl2




//// zero line cross (standard code)
ao = ta.sma(hl2,5) - ta.sma(hl2,34)
diff = ao - ao[1]


ao_long = bool(na)
ao_short = bool(na)

if aotype == "AC Zero Line Cross"
    ao_long := nRes > nRes[1] and nRes > 0
    ao_short := nRes < nRes[1] and nRes < 0
else if aotype == "AC Momentum Bar"
    ao_long := nRes > nRes[1]
    ao_short := nRes < nRes[1]
else if aotype == "Zero Line Cross"
    ao_long := ao > 0
    ao_short := ao < 0




////////////////////////////////////////////
///// WolfPack ID
////////////////////////////////////////////

wolfgroup = "██████████ Wolf Pack ID ██████████"
input1 = input(title='Fast Length', group=wolfgroup,defval=3)
input2 = input(title='Slow Length',group=wolfgroup, defval=8)
pivR = input(title='Wolfpack Wave Pivot Lookback Right', group=wolfgroup,defval=1)
pivL = input(title='Wolfpack Wave Pivot Lookback Left',group=wolfgroup, defval=15)
fastmaa = ta.ema(close, input1)
fastmab = ta.ema(close, input2)
wolfsrc = close
bspread = (fastmaa - fastmab) * 1.001
adline = 0
m = bspread > 0 ? color.new(color.lime, 0) : color.new(color.red, 0)
wolfup = ta.rma(math.max(ta.change(wolfsrc), 0), 3)
wolfdown = ta.rma(-math.min(ta.change(wolfsrc), 0), 3)
lbR = input(title='Divergence Pivot Lookback Right',group=wolfgroup, defval=1)
lbL = input(title='Divergence Pivot Lookback Left', group=wolfgroup,defval=10)
rangeUpper = input(title='Max of Lookback Range',group=wolfgroup, defval=100)
rangeLower = input(title='Min of Lookback Range', group=wolfgroup,defval=2)


osc = bspread
```

```
_inRange(cond) =>
    bars = ta.barssince(cond == true)
    rangeLower <= bars and bars <= rangeUpper



wolf_long = bspread > 0
wolf_short = bspread < 0




//////////////////////////////////////////
///// Bollinger Band (BB)
//////////////////////////////////////////

bb_group= "████████ Bollinger Band ████████"
bb_length = input.int(20, minval=1)
bb_maType = input.string("SMA", "Basis MA Type", options = ["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"]
bbsrc = input(close, title="Source")
bbmult = input.float(2.0, minval=0.001, maxval=50, title="StdDev")



bb_basis = ma(bbsrc, bb_length, bb_maType)
bbdev = bbmult * ta.stdev(bbsrc, bb_length)
bbupper = bb_basis + bbdev
bblower = bb_basis - bbdev
bboffset = input.int(0, "Offset", minval = -500, maxval = 500)
plot(switch_bb ? bb_basis:na, "Basis", color=#FF6D00, offset = bboffset)
bbp1 = plot(switch_bb ? bbupper:na, "Upper", color=#2962FF, offset = bboffset)
bbp2 = plot(switch_bb ? bblower:na, "Lower", color=#2962FF, offset = bboffset)
fillColor = switch_bb ? color.rgb(33, 150, 243, 95) : na
fill(bbp1, bbp2, title = "Background", color=fillColor)




//////////////////////////////////////////
///// BB Oscillator
//////////////////////////////////////////
bbgroup = "████████ Bollinger Band (BB) Oscillator ████████"

bbosc_length = input.int(20, minval=1,group=bbgroup)
bbosc_src = input(close, title='Source',group=bbgroup)
bbosc_mult = input.float(2.0, minval=0.001, maxval=50, title='StdDev',group=bbgroup)
bbosc_basis = ta.sma(bbosc_src, bbosc_length)
dlength = input.int(4, minval=1, title='Trigger Length',group=bbgroup)
bbosc_offset = input.int(0, 'Offset', minval=-500, maxval=500,group=bbgroup, tooltip = "Use Offset anc
bbosc_last = input(0, 'Show Last',group=bbgroup)
bbosc_dev = bbosc_mult * ta.stdev(bbosc_src, bbosc_length)
bbosc_upper = bbosc_basis + bbosc_dev
```

```
bbosc_lower = bbosc_basis - bbosc_dev
upercent = (bbosc_upper - close) / (bbosc_upper + close / 2)
lpercent = (bbosc_lower - close) / (bbosc_lower + close / 2)
bpercent = (bbosc_basis - close) / (bbosc_basis + close / 2)
usmooth = ta.wma(upercent, 6)
lsmooth = ta.wma(lpercent, 6)
bsmooth = ta.wma(bpercent, 6)
d1 = ta.sma(bsmooth, 2)
j = (bsmooth + d1) * -1
d2 = ta.sma(j, dlength)



bbosc_long = bool(na)
bbosc_short = bool(na)


bbcycle = 0
bbup = ta.crossover(j, usmooth)
bbdown = ta.crossunder(j, lsmooth)
bbcycle := bbup ? 1 : bbdown ? -1 : bbcycle[1]

if bbtype == "Entering Lower/Upper Band"
    bbosc_long := j > d2 and (j==lsmooth or j>lsmooth) and bbcycle==-1
    bbosc_short:= j < d2 and (j==usmooth or j<usmooth) and bbcycle==1
else if bbtype == "Exiting Lower/Upper Band"
    bbosc_long := j > d2 and (j>usmooth)
    bbosc_short:= j < d2 and (j<lsmooth)







//////////////////////////////////////////
///// Trend Meter
//////////////////////////////////////////
tm_group = "██████████ Trend Meter ██████████"

ShowTrendBar = true

WTSetups = input.bool(true, 'Wave Trend Filtered by Trend', group=tm_group, inline = 'tm')

TMSetups = input.bool(true, 'All 3 Trend Meters Now Align', group=tm_group, inline = 'tm2')

MSBar1 = 'Trend Filter'  // input(title= "1 - Wave Trend Signals",    defval = "Trend Filter",


MSBar2 = 'Trend Filter'  // input(title= "2 - Wave Trend Signals",    defval = "Filter X",



TrendBar1 = input.string(title='Trend Meter 1', defval='MACD Crossover - Fast - 8, 21, 5', options=['M

TrendBar2 = input.string(title='Trend Meter 2', defval='RSI 13: > or < 50', options=['MACD Crossover -
```

```
TrendBar3 = input.string(title='Trend Meter 3', defval='RSI 5: > or < 50', options=['MACD Crossover -

TrendBar4 = input.string(title='Trend Bar 1', defval='MA Crossover', options=['MA Crossover', 'MA Dire

TrendBar5 = input.string(title='Trend Bar 2', defval='MA Crossover', options=['MA Crossover', 'MA Dire


///////////////Signals - Wave Trend//////////////////////////////////////////////////////////////////


// Wave Trend - RSI

RSIMC = ta.rsi(close, 14)

// Wave Trend

ap = hlc3  // input(hlc3, "Wave Trend - Source")
n1 = 9  //input(9,    "Wave Trend - WT Channel Length")
n2 = 12  // input(12,   "Wave Trend - WT Average Length")
esa = ta.ema(ap, n1)
de = ta.ema(math.abs(ap - esa), n1)
ci = (ap - esa) / (0.015 * de)
tci = ta.ema(ci, n2)
wt11 = tci
wt22 = ta.sma(wt11, 3)

// Wave Trend - Overbought & Oversold lines

obLevel2 = 60  // input( 60,  "Wave Trend - WT Very Overbought")
obLevel = 50  // input( 50,  "Wave Trend - WT Overbought")
osLevel = -50  // input(-50,  "Wave Trend - WT Oversold")
osLevel2 = -60  // input(-60,  "Wave Trend - WT Very Oversold")

// Wave Trend - Conditions

WTCross = ta.cross(wt11, wt22)
WTCrossUp = wt22 - wt11 <= 0
WTCrossDown = wt22 - wt11 >= 0
WTOverSold = wt22 <= osLevel2
WTOverBought = wt22 >= obLevel2


// MA Inputs

MA1_Length = input.int(5, title='Fast MA', minval=1, group='Trend Bar 1 - Settings', inline='TB1 Fast'
MA1_Type = input.string(title='', defval='EMA', options=['EMA', 'SMA'], group='Trend Bar 1 - Settings'

MA2_Length = input.int(11, title='Slow MA', minval=1, group='Trend Bar 1 - Settings', inline='TB1 Slow
MA2_Type = input.string(title='', defval='EMA', options=['EMA', 'SMA'], group='Trend Bar 1 - Settings'

MA3_Length = input.int(9, title='Fast MA', minval=1, group='Trend Bar 2 - Settings', inline='TB2 Fast'
MA3_Type = input.string(title='', defval='EMA', options=['EMA', 'SMA'], group='Trend Bar 2 - Settings'

MA4_Length = input.int(21, title='Slow MA', minval=1, group='Trend Bar 2 - Settings', inline='TB2 Slow
```

```
MA4_Type = input.string(title='', defval='SMA', options=['EMA', 'SMA'], group='Trend Bar 2 - Settings'


// MA Calculations

Close = request.security(syminfo.tickerid, timeframe.period, close, lookahead=barmerge.lookahead_on)


MA1 = if MA1_Type == 'SMA'
    ta.sma(Close, MA1_Length)
else
    ta.ema(Close, MA1_Length)


MA2 = if MA2_Type == 'SMA'
    ta.sma(Close, MA2_Length)
else
    ta.ema(Close, MA2_Length)


MA3 = if MA3_Type == 'SMA'
    ta.sma(Close, MA3_Length)
else
    ta.ema(Close, MA3_Length)


MA4 = if MA4_Type == 'SMA'
    ta.sma(Close, MA4_Length)
else
    ta.ema(Close, MA4_Length)


// MA Crossover Condition

MACrossover1 = MA1 > MA2 ? 1 : 0

MACrossover2 = MA3 > MA4 ? 1 : 0

// MA Direction Condition

MA1Direction = MA1 > MA1[1] ? 1 : 0

MA2Direction = MA2 > MA2[1] ? 1 : 0

MA3Direction = MA3 > MA3[1] ? 1 : 0

MA4Direction = MA4 > MA4[1] ? 1 : 0

// MA Direction Change Condition

MA1PositiveDirectionChange = MA1Direction and not MA1Direction[1] ? 1 : 0

MA2PositiveDirectionChange = MA2Direction and not MA2Direction[1] ? 1 : 0

MA3PositiveDirectionChange = MA3Direction and not MA3Direction[1] ? 1 : 0
```

```
MA4PositiveDirectionChange = MA4Direction and not MA4Direction[1] ? 1 : 0


MA1NegativeDirectionChange = not MA1Direction and MA1Direction[1] ? 1 : 0

MA2NegativeDirectionChange = not MA2Direction and MA2Direction[1] ? 1 : 0

MA3NegativeDirectionChange = not MA3Direction and MA3Direction[1] ? 1 : 0

MA4NegativeDirectionChange = not MA4Direction and MA4Direction[1] ? 1 : 0


// MACD and MOM & DAD - Top Dog Trading

// Standard MACD Calculations

MACDfastMA = 12
MACDslowMA = 26
MACDsignalSmooth = 9


MACDLine = ta.ema(close, MACDfastMA) - ta.ema(close, MACDslowMA)

SignalLine = ta.ema(MACDLine, MACDsignalSmooth)

MACDHistogram = MACDLine - SignalLine


// MACD- Background Color Change Condition

MACDHistogramCross = MACDHistogram > 0 ? 1 : 0

MACDLineOverZero = MACDLine > 0 ? 1 : 0

MACDLineOverZeroandHistogramCross = MACDHistogramCross and MACDLineOverZero ? 1 : 0

MACDLineUnderZeroandHistogramCross = not MACDHistogramCross and not MACDLineOverZero ? 1 : 0


// Fast MACD Calculations

FastMACDfastMA = 8
FastMACDslowMA = 21
FastMACDsignalSmooth = 5


FastMACDLine = ta.ema(close, FastMACDfastMA) - ta.ema(close, FastMACDslowMA)

FastSignalLine = ta.ema(FastMACDLine, FastMACDsignalSmooth)

FastMACDHistogram = FastMACDLine - FastSignalLine

// Fast MACD- Background Color Change Condition
```

```
FastMACDHistogramCross = FastMACDHistogram > 0 ? 1 : 0

FastMACDLineOverZero = FastMACDLine > 0 ? 1 : 0

FastMACDLineOverZeroandHistogramCross = FastMACDHistogramCross and FastMACDLineOverZero ? 1 : 0

FastMACDLineUnderZeroandHistogramCross = not FastMACDHistogramCross and not FastMACDLineOverZero ? 1 :


// Top Dog Trading - Mom Dad Calculations

TopDog_Fast_MA = 5
TopDog_Slow_MA = 20
TopDog_Sig = 30


TopDogMom = ta.ema(close, TopDog_Fast_MA) - ta.ema(close, TopDog_Slow_MA)

TopDogDad = ta.ema(TopDogMom, TopDog_Sig)

// Top Dog Dad - Background Color Change Condition

TopDogDadDirection = TopDogDad > TopDogDad[1] ? 1 : 0

TopDogMomOverDad = TopDogMom > TopDogDad ? 1 : 0

TopDogMomOverZero = TopDogMom > 0 ? 1 : 0




////// Trend Barmeter Calculations //////


haclose_tm = ohlc4
haopen_tm = 0.0
haopen_tm := na(haopen_tm[1]) ? (open + close) / 2 : (haopen_tm[1] + haclose_tm[1]) / 2



// RSI 5 Trend Barmeter Calculations

RSI5 = ta.rsi(close, 5)

RSI5Above50 = RSI5 > 50 ? 1 : 0



// RSI 5 Trend Barmeter Calculations

RSI13 = ta.rsi(close, 13)


// Linear Regression Calculation For RSI Signal Line
```

```
SignalLineLength1 = 21

x = bar_index
y = RSI13
x_ = ta.sma(x, SignalLineLength1)
y_ = ta.sma(y, SignalLineLength1)
mx = ta.stdev(x, SignalLineLength1)
my = ta.stdev(y, SignalLineLength1)
c = ta.correlation(x, y, SignalLineLength1)
slope = c * (my / mx)
inter = y_ - slope * x_
LinReg1 = x * slope + inter


RSISigDirection = LinReg1 > LinReg1[1] ? 1 : 0

RSISigCross = RSI13 > LinReg1 ? 1 : 0

RSI13Above50 = RSI13 > 50 ? 1 : 0




TrendBar1Result = TrendBar1 == 'MA Crossover' ? MACrossover1 : TrendBar1 == 'MACD Crossover - 12, 26,

TrendBar2Result = TrendBar2 == 'MA Crossover' ? MACrossover1 : TrendBar2 == 'MACD Crossover - 12, 26,

TrendBar3Result = TrendBar3 == 'MA Crossover' ? MACrossover1 : TrendBar3 == 'MACD Crossover - 12, 26,



FilterXUp = FastMACDHistogramCross and ta.ema(close, 15) > ta.ema(close, 15)[1]

FilterXDown = not FastMACDHistogramCross and ta.ema(close, 15) < ta.ema(close, 15)[1]


TrendFilterPlus = ta.ema(close, 15) > ta.ema(close, 20) and ta.ema(close, 20) > ta.ema(close, 30) and

TrendFilterMinus = ta.ema(close, 15) < ta.ema(close, 20) and ta.ema(close, 20) < ta.ema(close, 30) and



MSBar1PositiveWaveTrendSignal = MSBar1 == 'Filter X' ? FilterXUp and WTCross and WTCrossUp : MSBar1 ==
MSBar1NegativeWaveTrendSignal = MSBar1 == 'Filter X' ? FilterXDown and WTCross and WTCrossDown : MSBar

MSBar2PositiveWaveTrendSignal = MSBar2 == 'Filter X' ? FilterXUp and WTCross and WTCrossUp : MSBar2 ==
MSBar2NegativeWaveTrendSignal = MSBar2 == 'Filter X' ? FilterXDown and WTCross and WTCrossDown : MSBar


////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
CrossoverType2 = TrendBar4 == 'DAD Direction (Top Dog Trading)' ? TopDogDadDirection : TrendBar4 == 'M

CrossoverType3 = TrendBar5 == 'DAD Direction (Top Dog Trading)' ? TopDogDadDirection : TrendBar5 == 'M




MaxValueMACrossUp = ta.crossover(ta.ema(Close, 5), ta.ema(Close, 11))
MaxValueMACrossDown = ta.crossunder(ta.ema(Close, 5), ta.ema(Close, 11))

TB1MACrossUp = ta.crossover(MA1, MA2)
TB1MACrossDown = ta.crossunder(MA1, MA2)



TB2MACrossUp = ta.crossover(MA3, MA4)
TB2MACrossDown = ta.crossunder(MA3, MA4)


TB1Green = MA1 > MA2
TB1Red = MA1 < MA2

TB2Green = MA3 > MA4
TB2Red = MA3 < MA4

TB12Green = TB1Green and TB2Green and (TB1MACrossUp or TB2MACrossUp)
TB12Red = TB1Red and TB2Red and (TB1MACrossDown or TB2MACrossDown)



////////////////////////////////
/////// Stochastic
////////////////////////////////
groupname = "██████████ Stochastic ██████████"
len = input.int(14, minval=1, title="Length",group=groupname)
smoothK = input.int(3, minval=1, title="K Smoothing",group=groupname)
smoothD = input.int(3, minval=1, title="D Smoothing",group=groupname)
upLine = input.int(80, minval=50, maxval=90, title="Overbought level",group=groupname)
lowLine = input.int(20, minval=10, maxval=50, title="Oversold level",group=groupname)

//Resolutioon for MTF
resstoch = timeframe.period
//Stoch formula
kk = ta.sma(ta.stoch(close, high, low, len), smoothK)
dd = ta.sma(kk, smoothD)
outK = request.security(syminfo.tickerid, resstoch, kk)
outD = request.security(syminfo.tickerid, resstoch, dd)

//definitions for Cross
aboveLine = outK > upLine ? 1 : 0
```

```
belowLine = outK < lowLine ? 1 : 0
stoch_long = bool (na)
stoch_short = bool (na)
if stochtype == "CrossOver"
    stoch_long := (outK[1] < outD[1] and outK > outD) ? 1 : 0
    stoch_short := (outK[1] > outD[1] and outK < outD) ? 1 : 0
else if stochtype == "CrossOver in OB & OS levels"
    stoch_long := (outK[1] < outD[1] and outK[1] < lowLine[1]) and (outK > outD) and outK > lowLine? 1
    stoch_short := (outK[1] > outD[1] and outK[1] > upLine[1]) and (outK < outD) and outK < upLine? 1
else if stochtype == "%K above/below %D"
    stoch_long := outK > outD
    stoch_short := outK < outD




////////////////////////////////////////////
///////RSI
////////////////////////////////////////////

rsi_group = "▒▒▒▒▒▒▒▒▒▒▒▒▒▒ RSI  ▒▒▒▒▒▒▒▒▒▒▒▒▒▒"

rsiLengthInput = input.int(14, minval=1, title="RSI Length", group=rsi_group)
rsiSourceInput = input.source(close, "Source", group=rsi_group)
maTypeInput = input.string("SMA", title="MA Type", options=["SMA", "Bollinger Bands", "EMA", "SMMA (RM
rsi_upper = input.int(defval=80, title='Overbought Zone', group=rsi_group, inline='zone')
rsi_lower = input.int(defval=20, title='Oversold Zone', group=rsi_group, inline='zone')

respectrsilevel = input.int(defval=50, minval=1, title='RSI MidLine', group=rsi_group)
maLengthInput = input.int(14, title="MA Length", group=rsi_group)

up = ta.rma(math.max(ta.change(rsiSourceInput), 0), rsiLengthInput)
down = ta.rma(-math.min(ta.change(rsiSourceInput), 0), rsiLengthInput)
rsi = down == 0 ? 100 : up == 0 ? 0 : 100 - (100 / (1 + up / down))
rsiMA = ma(rsi, maLengthInput, maTypeInput)
isBB = maTypeInput == "Bollinger Bands"




////////////////////////////////////////////
/// HULL SUITE
////////////////////////////////////////////
hull_group = "▒▒▒▒▒▒▒▒▒ HullSuite ▒▒▒▒▒▒▒▒▒"
//INPUT
hullsrc = input(close, title='Source',group=hull_group)
modeSwitch = input.string('Hma', title='Hull Variation', options=['Hma', 'Thma', 'Ehma'],group=hull_gr
hull_length = input(55, title='hull_length(180-200 for floating S/R , 55 for swing entry)',group=hull_
hull_lengthMult = input(1.0, title='hull_length multiplier (Used to view higher timeframes with straig

useHtf = input(false, title='Show Hull MA from X timeframe? (good for scalping)',group=hull_group)
htf = input.timeframe('240', title='Higher timeframe',group=hull_group)
```

```
//FUNCTIONS
//HMA
HMA(_hullsrc, _hull_length) =>
    ta.wma(2 * ta.wma(_hullsrc, _hull_length / 2) - ta.wma(_hullsrc, _hull_length), math.round(math.sc
//EHMA
EHMA(_hullsrc, _hull_length) =>
    ta.ema(2 * ta.ema(_hullsrc, _hull_length / 2) - ta.ema(_hullsrc, _hull_length), math.round(math.sc
//THMA
THMA(_hullsrc, _hull_length) =>
    ta.wma(ta.wma(_hullsrc, _hull_length / 3) * 3 - ta.wma(_hullsrc, _hull_length / 2) - ta.wma(_hulls

//SWITCH
Mode(modeSwitch, hullsrc, len) =>
    modeSwitch == 'Hma' ? HMA(hullsrc, len) : modeSwitch == 'Ehma' ? EHMA(hullsrc, len) : modeSwitch =

//OUT
_hull = Mode(modeSwitch, hullsrc, int(hull_length * hull_lengthMult))
HULL = useHtf ? request.security(syminfo.ticker, htf, _hull) : _hull
MHULL = HULL[0]
SHULL = HULL[2]

//COLOR
// hullColor = switchColor ? HULL > HULL[2] ? #00ff00 : #ff0000 : #ff9800



//////////////////////////
/// STC overlay signal
//////////////////////////
stc_group = "██████████ Schaff Trend Cycle (STC) ██████████"
fastLength = input(title='MACD Fast Length', defval=23, group=stc_group)
slowLength = input(title='MACD Slow Length', defval=50, group=stc_group)
cycleLength = input(title='Cycle Length', defval=10, group=stc_group)
d1Length = input(title='1st %D Length', defval=3, group=stc_group)
d2Length = input(title='2nd %D Length', defval=3, group=stc_group)
srcstc = input(title='Source', defval=close, group=stc_group)
upper = input(title='Upper Band', defval=75, group=stc_group)
lower = input(title='Lower Band', defval=25, group=stc_group)
v_show_last  = input(2000,  "Plotting Length", group=stc_group)

macd = ta.ema(srcstc, fastLength) - ta.ema(srcstc, slowLength)
k = nz(fixnan(ta.stoch(macd, macd, macd, cycleLength)))
d = ta.ema(k, d1Length)
kd = nz(fixnan(ta.stoch(d, d, d, cycleLength)))
stc = ta.ema(kd, d2Length)
stc := math.max(math.min(stc, 100), 0)

stcColor1 = stc > stc[1] ? color.green : color.red
stcColor2 = stc > upper ? color.green : stc <= lower ? color.red : color.orange


upperCrossover = ta.crossover(stc, upper)
upperCrossunder = ta.crossunder(stc, upper)
lowerCrossover = ta.crossover(stc, lower)
lowerCrossunder = ta.crossunder(stc, lower)
```

```
stcup = stc >= upper
stcdown = stc <= lower


plotshape(stcdown and switch_stc? true :na, style=shape.circle, location=location.top , show_last = v_
plotshape(stcup and switch_stc? true:na, style=shape.circle, location=location.top, show_last = v_show



////////////////////////////////////////////////////
//vector candles
////////////////////////////////////////////////////

// Indicator Settings

pvsra_group="[redacted] PVSRA [redacted]"

var overideCandleColours = input.bool(title='Override Candles with PVSRA Colour', defval=true, tooltip

var bool override_imnt = input.bool(defval=false, title="Overide Symbol", group=pvsra_group, inline="0
var string pvsra_sym = input.symbol(title="", defval="BINANCE:BTCUSDTPERP", group=pvsra_group, inline=

var Bull200CandleColor = input.color(color.new(color.lime, 0), title="200% Volume", group=pvsra_group,
var Bear200CandleColor = input.color(color.new(color.red, 0), title="", group=pvsra_group, inline = "1

var Bull150CandleColor = input.color(color.new(color.blue, 0), title="150% Volume", group=pvsra_group,
var Bear150CandleColor = input.color(color.new(color.fuchsia, 0), title="", group=pvsra_group, inline=

var BullNormCandleColor = input.color(color.new(#999999, 0), title="Norm Volume", group=pvsra_group, i
var BearNormCandleColor = input.color(color.new(#4d4d4d, 0), title="", group=pvsra_group, inline="3")


var color candleColor = na
var color imbalanceColor = na
var color imbalancedLineColor = na

var color NO_COLOR = na


var bool chartIs120MinOrMore = false



// Logic to reference another Instruments Volume Profile


pvsra_imnt(sresolution,sseries) => request.security(override_imnt ? pvsra_sym : syminfo.tickerid ,sres
volume_imnt = override_imnt == true? pvsra_imnt("",volume): volume
high_imnt = override_imnt == true? pvsra_imnt("",high): high
low_imnt = override_imnt == true? pvsra_imnt("",low): low
close_imnt = override_imnt == true? pvsra_imnt("",close): close
open_imnt = override_imnt == true? pvsra_imnt("",open): open
```

```
av = ta.sma(volume_imnt, 10)//sum_2 = math.sum(volume, 10)
value2 = volume_imnt * (high_imnt - low_imnt)
hivalue2 = ta.highest(value2, 10)
imnt_override_pvsra_calc_part2 = volume_imnt >= av * 1.5 ? 2 : 0
va = volume_imnt >= av * 2 or value2 >= hivalue2 ? 1 : imnt_override_pvsra_calc_part2


// Bull or bear Candle Colors
isBull =  close_imnt > open_imnt


var bool is200Bull = na
var bool is150Bull = na
var bool is100Bull = na
var bool is200Bear = na
var bool is150Bear = na
var bool is100Bear = na


if isBull
    if va == 1
        candleColor := Bull200CandleColor
        is200Bull := true
    else
        if va == 2
            candleColor := Bull150CandleColor
            is150Bull := true
        else
            is200Bull := false
            is150Bull := false
            candleColor := BullNormCandleColor
            imbalanceColor := na
            imbalancedLineColor := na
else
    if va == 1
        candleColor := Bear200CandleColor
        is200Bear := true
    else
        if va == 2
            candleColor := Bear150CandleColor
            is150Bear := true
        else
            is200Bear := false
            is150Bear := false
            candleColor := BearNormCandleColor
            imbalanceColor := na
            imbalancedLineColor := na

barcolor(overideCandleColours and switch_pvsra ? candleColor : NO_COLOR)
plotcandle(open, high, low, close, color=(overideCandleColours and switch_pvsra ? candleColor : NO_COL


////////////////////////////////////////////////
//////// SUpply/Demand POI
```

```
////////////////////////////////////////////////
//      INDICATOR SETTINGS
poi_group = '██████████ Supply/Demand Zone ██████████'
swing_length = input.int(10, title = 'Swing High/Low Length', group = poi_group, minval = 1, maxval =
history_of_demand_to_keep = input.int(20, title = 'History To Keep', minval = 5, maxval = 50, group =
box_width = input.float(2.5, title = 'Supply/Demand Box Width', group = poi_group, minval = 1, maxval

//      INDICATOR VISUAL SETTINGS
show_zigzag = input.bool(false, title = 'Show Zig Zag', group = 'Visual Settings', inline = '1')
show_price_action_labels = input.bool(false, title = 'Show Price Action Labels', group = 'Visual Setti

supply_color = input.color(color.new(#EDEDED,70), title = 'Supply', group = 'Visual Settings', inline
supply_outline_color = input.color(color.new(color.white,100), title = 'Outline', group = 'Visual Sett

demand_color = input.color(color.new(#00FFFF,70), title = 'Demand', group = 'Visual Settings', inline
demand_outline_color = input.color(color.new(color.white,100), title = 'Outline', group = 'Visual Sett


poi_label_color = input.color(color.white, title = 'POI Label', group = 'Visual Settings', inline = '7

swing_type_color = input.color(color.black, title = 'Price Action Label', group = 'Visual Settings', i
zigzag_color = input.color(color.new(#000000,0), title = 'Zig Zag', group = 'Visual Settings', inline

//
//END SETTINGS
//

atrpoi = ta.atr(50)
//
//FUNCTIONS
//

//      FUNCTION TO ADD NEW AND REMOVE LAST IN ARRAY
f_array_add_pop(array, new_value_to_add) =>
    array.unshift(array, new_value_to_add)
    array.pop(array)

//      FUNCTION SWING H & L LABELS
f_sh_sl_labels(array, swing_type) =>

    var string label_text = na
    if swing_type == 1
        if array.get(array, 0) >= array.get(array, 1)
            label_text := 'HH'
        else
            label_text := 'LH'
        label.new(bar_index - swing_length, array.get(array,0), text = label_text, style=label.style_l

    else if swing_type == -1
        if array.get(array, 0) >= array.get(array, 1)
            label_text := 'HL'
        else
            label_text := 'LL'
        label.new(bar_index - swing_length, array.get(array,0), text = label_text, style=label.style_l
```

```
//      FUNCTION MAKE SURE SUPPLY ISNT OVERLAPPING
f_check_overlapping(new_poi, box_array, atrpoi) =>

    atr_threshold = atrpoi * 2
    okay_to_draw = true

    for i = 0 to array.size(box_array) - 1
        top = box.get_top(array.get(box_array, i))
        bottom = box.get_bottom(array.get(box_array, i))
        poi = (top + bottom) / 2

        upper_boundary = poi + atr_threshold
        lower_boundary = poi - atr_threshold

        if new_poi >= lower_boundary and new_poi <= upper_boundary
            okay_to_draw := false
            break
        else
            okay_to_draw := true
    okay_to_draw


//      FUNCTION TO DRAW SUPPLY OR DEMAND ZONE
f_supply_demand(value_array, bn_array, box_array, label_array, box_type, atrpoi) =>

    atr_buffer = atrpoi * (box_width / 10)
    box_left = array.get(bn_array, 0)
    box_right = bar_index

    var float box_top = 0.00
    var float box_bottom = 0.00
    var float poi = 0.00


    if box_type == 1
        box_top := array.get(value_array, 0)
        box_bottom := box_top - atr_buffer
        poi := (box_top + box_bottom) / 2
    else if box_type == -1
        box_bottom := array.get(value_array, 0)
        box_top := box_bottom + atr_buffer
        poi := (box_top + box_bottom) / 2

    okay_to_draw = f_check_overlapping(poi, box_array, atrpoi)
    // okay_to_draw = true

    //delete oldest box, and then create a new box and add it to the array
    if box_type == 1 and okay_to_draw and switch_poi
        box.delete( array.get(box_array, array.size(box_array) - 1) )
        f_array_add_pop(box_array, box.new( left = box_left, top = box_top, right = box_right, bottom
             bgcolor = supply_color, extend = extend.right, text = 'SUPPLY', text_halign = text.align_

        box.delete( array.get(label_array, array.size(label_array) - 1) )
        f_array_add_pop(label_array, box.new( left = box_left, top = poi, right = box_right, bottom =
             bgcolor = color.new(poi_label_color,90), extend = extend.right, text = 'POI', text_haligr
```

```
    else if box_type == -1 and okay_to_draw  and switch_poi
        box.delete( array.get(box_array, array.size(box_array) - 1) )
        f_array_add_pop(box_array, box.new( left = box_left, top = box_top, right = box_right, bottom
             bgcolor = demand_color, extend = extend.right,  text = 'DEMAND', text_halign = text.aligr

        box.delete( array.get(label_array, array.size(label_array) - 1) )
        f_array_add_pop(label_array, box.new( left = box_left, top = poi, right = box_right, bottom =
             bgcolor = color.new(poi_label_color,90), extend = extend.right,  text = 'POI', text_halig


//      FUNCTION TO CHANGE SUPPLY/DEMAND TO A BOS IF BROKEN
f_sd_to_bos(box_array, bos_array, label_array, zone_type) =>

    if zone_type == 1  and switch_poi
        for i = 0 to array.size(box_array) - 1
            level_to_break = box.get_top(array.get(box_array,i))
            // if ta.crossover(close, level_to_break)
            if close >= level_to_break
                copied_box = box.copy(array.get(box_array,i))
                f_array_add_pop(bos_array, copied_box)
                mid = (box.get_top(array.get(box_array,i)) + box.get_bottom(array.get(box_array,i))) /
                box.set_top(array.get(bos_array,0), mid)
                box.set_bottom(array.get(bos_array,0), mid)
                box.set_extend( array.get(bos_array,0), extend.none)
                box.set_right( array.get(bos_array,0), bar_index)
                box.set_text( array.get(bos_array,0), '' )
                box.set_text_color( array.get(bos_array,0), color.new(color.white, 0))
                box.set_text_size( array.get(bos_array,0), size.small)
                box.set_text_halign( array.get(bos_array,0), text.align_center)
                box.set_text_valign( array.get(bos_array,0), text.align_center)
                box.delete(array.get(box_array, i))
                box.delete(array.get(label_array, i))


    if zone_type == -1  and switch_poi
        for i = 0 to array.size(box_array) - 1
            level_to_break = box.get_bottom(array.get(box_array,i))
            // if ta.crossunder(close, level_to_break)
            if close <= level_to_break
                copied_box = box.copy(array.get(box_array,i))
                f_array_add_pop(bos_array, copied_box)
                mid = (box.get_top(array.get(box_array,i)) + box.get_bottom(array.get(box_array,i))) /
                box.set_top(array.get(bos_array,0), mid)
                box.set_bottom(array.get(bos_array,0), mid)
                box.set_extend( array.get(bos_array,0), extend.none)
                box.set_right( array.get(bos_array,0), bar_index)
                box.set_text( array.get(bos_array,0), '' )
                box.set_text_color( array.get(bos_array,0),  color.new(color.white, 0))
                box.set_text_size( array.get(bos_array,0), size.small)
                box.set_text_halign( array.get(bos_array,0), text.align_center)
                box.set_text_valign( array.get(bos_array,0), text.align_center)
                box.delete(array.get(box_array, i))
                box.delete(array.get(label_array, i))
```

```
//      FUNCTION MANAGE CURRENT BOXES BY CHANGING ENDPOINT
f_extend_box_endpoint(box_array) =>

    for i = 0 to array.size(box_array) - 1
        box.set_right(array.get(box_array, i), bar_index + 100)




//      CALCULATE SWING HIGHS & SWING LOWS
swing_high = ta.pivothigh(high, swing_length, swing_length)
swing_low = ta.pivotlow(low, swing_length, swing_length)

//      ARRAYS FOR SWING H/L & BN
var swing_high_values = array.new_float(5,0.00)
var swing_low_values = array.new_float(5,0.00)

var swing_high_bns = array.new_int(5,0)
var swing_low_bns = array.new_int(5,0)

//      ARRAYS FOR SUPPLY / DEMAND
var current_supply_box = array.new_box(history_of_demand_to_keep, na)
var current_demand_box = array.new_box(history_of_demand_to_keep, na)

//      ARRAYS FOR SUPPLY / DEMAND POI LABELS
var current_supply_poi = array.new_box(history_of_demand_to_keep, na)
var current_demand_poi = array.new_box(history_of_demand_to_keep, na)

//      ARRAYS FOR BOS
var supply_bos = array.new_box(5, na)
var demand_bos = array.new_box(5, na)
//
//END CALCULATIONS
//

//      NEW SWING HIGH
if not na(swing_high)

    //MANAGE SWING HIGH VALUES
    f_array_add_pop(swing_high_values, swing_high)
    f_array_add_pop(swing_high_bns, bar_index[swing_length])
    if show_price_action_labels
        f_sh_sl_labels(swing_high_values, 1)

    f_supply_demand(swing_high_values, swing_high_bns, current_supply_box, current_supply_poi, 1, atrp

//      NEW SWING LOW
else if not na(swing_low)

    //MANAGE SWING LOW VALUES
    f_array_add_pop(swing_low_values, swing_low)
    f_array_add_pop(swing_low_bns, bar_index[swing_length])
```

```
    if show_price_action_labels
        f_sh_sl_labels(swing_low_values, -1)

    f_supply_demand(swing_low_values, swing_low_bns, current_demand_box, current_demand_poi, -1, atrpc


f_sd_to_bos(current_supply_box, supply_bos, current_supply_poi, 1)
f_sd_to_bos(current_demand_box, demand_bos, current_demand_poi, -1)

f_extend_box_endpoint(current_supply_box)
f_extend_box_endpoint(current_demand_box)

//ZIG ZAG
h = ta.highest(high, swing_length * 2 + 1)
l = ta.lowest(low, swing_length * 2 + 1)
f_isMin(len) =>
    l == low[len]
f_isMax(len) =>
    h == high[len]

var dirUp = false
var lastLow = high * 100
var lastHigh = 0.0
var timeLow = bar_index
var timeHigh = bar_index
var line li = na

f_drawLine() =>
    _li_color = show_zigzag and switch_poi ? zigzag_color : color.new(#ffffff,100)
    line.new(timeHigh - swing_length, lastHigh, timeLow - swing_length, lastLow, xloc.bar_index, color

if dirUp
    if f_isMin(swing_length) and low[swing_length] < lastLow
        lastLow := low[swing_length]
        timeLow := bar_index
        line.delete(li)
        li := f_drawLine()
        li

    if f_isMax(swing_length) and high[swing_length] > lastLow
        lastHigh := high[swing_length]
        timeHigh := bar_index
        dirUp := false
        li := f_drawLine()
        li

if not dirUp
    if f_isMax(swing_length) and high[swing_length] > lastHigh
        lastHigh := high[swing_length]
        timeHigh := bar_index
        line.delete(li)
        li := f_drawLine()
        li
    if f_isMin(swing_length) and low[swing_length] < lastHigh
        lastLow := low[swing_length]
```

```
        timeLow := bar_index
        dirUp := true
        li := f_drawLine()
        if f_isMax(swing_length) and high[swing_length] > lastLow
            lastHigh := high[swing_length]
            timeHigh := bar_index
            dirUp := false
            li := f_drawLine()
            li


////////////////////////////////////////////////
////////  Heiken Ashi Candle
////////////////////////////////////////////////


hagroup = "██████ Heiken-ashi candles ██████"


hkClose         = (open + high + low + close) / 4
hkOpen          = float(na)
hkOpen          := na(hkOpen[1]) ? (open + close) / 2 : (nz(hkOpen[1]) + nz(hkClose[1])) / 2
hkHigh          = math.max(high, math.max(hkOpen, hkClose))
hkLow           = math.min(low,  math.min(hkOpen, hkClose))

//[hkOpen, hkHigh, hkLow, hkClose] = security(heikinashi(syminfo.tickerid), timeframe.period, [open, h

candletype      = input.string  ("Hollow",      "Candle Type",  options=["Hollow", "Bars", "Candles"],

BodyBull        = input.color   (#26a69a,       "",             inline="a",         group="Candle Body
BodyBear        = input.color   (#ef5350,       "",             inline="a",         group="Candle Body
BorderBull      = input.color   (#26a69a,       "",             inline="b",         group="Candle Borc
BorderBear      = input.color   (#ef5350,       "",             inline="b",         group="Candle Borc
WickBull        = input.color   (#26a69a,       "",             inline="c",         group="Candle Wick
WickBear        = input.color   (#ef5350,       "",             inline="c",         group="Candle Wick

hollow          = candletype == "Hollow"
bars            = candletype == "Bars"
candle          = candletype == "Candles"


plotcandle(
  hkOpen, hkHigh, hkLow, hkClose,
  "Hollow Candles",
  switch_ha ? (hollow ? hkClose < hkOpen ? BodyBear : na : candle ? hkClose < hkOpen ? BodyBear : Body
  switch_ha ? (hollow or candle ? hkClose < hkOpen ? WickBear : WickBull : na) : na,
  bordercolor = switch_ha ? (hollow or candle ? hkClose < hkOpen ? BorderBear : BorderBull : na) : na)




////////////////////////////////////////////////
////////  Fair Value gap
```

```
////////////////////////////////////////////////

fvggroup = "████████ Fair Value Gap (FVG) ████████"
numDays = input.int(7, "number of days lookback",group=fvggroup)
showUP = input.bool(true, "'UP' FVGs:", inline ='1',group=fvggroup)
colUp = input.color(color.new(color.blue, 86), "", inline ='1',group=fvggroup)
showDN = input.bool(true, "'DOWN' FVGs:", inline ='2',group=fvggroup)
colDn = input.color(color.new(color.orange, 86), "", inline ='2',group=fvggroup)
showCE = input.bool(true, "show CE", inline ='3',group=fvggroup)
ceCol = input.color(color.new(color.black, 1), "| color:", inline ='3',group=fvggroup)
ceStyle = input.string(line.style_dotted, "| style:", options=[line.style_dotted,line.style_solid,  li
deleteFilledBoxes = input.bool(true, "delete filled boxes & lines",group=fvggroup)
CEcond = input.bool (false, "Use CE (as opposed to Full Fill)",group=fvggroup, tooltip = "If toggled C
colorNone = color.new(color.white, 100)
_day = 24*3600*1000
var box bxUp = na, var box bxDn = na, var line lnUp = na, var line lnDn = na
var array<box> bxUpArr = array.new<box>(0), var array<line> lnUpArr = array.new<line>(0)
var array<box> bxDnArr = array.new<box>(0), var array<line> lnDnArr = array.new<line>(0)
dnCE = high[1] + (low[3]-high[1])/2
upCE = low[1] - (low[1]-high[3])/2
if low[3] > high[1] and time> timenow- numDays*_day and showDN and switch_fvg
    bxDnArr.push(box.new(bar_index-3, low[3], bar_index, high[1], bgcolor = colDn, border_color = colc
    lnDnArr.push(line.new(bar_index-3, dnCE, bar_index, dnCE, color = showCE?ceCol:colorNone, style =c
if high[3] < low[1] and time> timenow- numDays*_day and showUP and switch_fvg
    bxUpArr.push(box.new(bar_index-3, low[1], bar_index, high[3], bgcolor = colUp, border_color = colc
    lnUpArr.push(line.new(bar_index-3, upCE, bar_index, upCE, color = showCE?ceCol:colorNone, style =c

var array<int> _countArr =array.new<int>(0)
var array<int> _countArrIOFED =array.new<int>(0)


    //modified form of @Bjorgum's looping function.  This stops boxes/lines painting when price passes
extendAndRemoveBx(array<box> boxArray, array<line> lineArray, array<int> countArr1, array<int> countAr
    if boxArray.size() > 0
        for i = boxArray.size() -1 to 0
            line ln = lineArray.get(i)
            box bx = boxArray.get(i)
            bx.set_right(bar_index)
            ln.set_x2(bar_index)
            float price = CEcond?ln.get_price(bar_index):(isBull?bx.get_top():bx.get_bottom())
            float price_IOFED = isBull?bx.get_bottom():bx.get_top()
            int m = isBull ? 1 : -1
            float hiLo = isBull ? high : low
            if hiLo * m > price * m
                boxArray.remove(i)
                lineArray.remove(i)
                countArr1.push(isBull?1:-1) //for 'above/below threshold alerts; counter sum will decr
                if deleteFilledBoxes
                    bx.set_bgcolor(colorNone)
                    ln.set_color(colorNone)
            if hiLo*m>price_IOFED*m
                countArr2.push(isBull?1:-1)

    if boxArray.size() > maxSize
```

```
        box.delete(boxArray.shift())
        line.delete(lineArray.shift())

extendAndRemoveBx(bxDnArr,lnDnArr,_countArr,_countArrIOFED, true, 12) //12 should be good for around 2
extendAndRemoveBx(bxUpArr, lnUpArr,_countArr,_countArrIOFED, false, 12)

upThresholdLst = array.sum(_countArr)>array.sum(_countArr)[1]
dnThresholdLst = array.sum(_countArr)<array.sum(_countArr)[1]

upIOFEDlast= array.sum(_countArrIOFED)>array.sum(_countArrIOFED)[1]
dnIOFEDlast= array.sum(_countArrIOFED)<array.sum(_countArrIOFED)[1]


////////////////////////////////////////////////
////////// Vector Zone
////////////////////////////////////////////////

import TradersReality/Traders_Reality_Lib/1 as trLib

vz_group = "██████████ Liquidity Zone ██████████"
color redVectorColor = input.color(title='Vector: Red', group=vz_group, defval=color.red, inline='vect
color greenVectorColor = input.color(title='Green', group=vz_group, defval=color.lime, inline='vectors
color violetVectorColor = input.color(title='Violet', group=vz_group, defval=color.fuchsia, inline='ve
color blueVectorColor = input.color(title='Blue', group=vz_group, defval=color.blue, inline='vectors',
color regularCandleUpColor = input.color(title='Regular: Up Candle', group=vz_group, defval=#999999, i
color regularCandleDownColor = input.color(title='Down Candle', group=vz_group, defval=#4d4d4d, inline
bool setcandlecolors = input.bool(false, title='Set PVSRA candle colors?', group=vz_group, inline='set

int zonesMax = input.int(500, 'Maximum zones to draw', group=vz_group)
string zoneType = input.string(group=vz_group, defval='Body only', title='Zone top/bottom is defined w
string zoneUpdateType = input.string(group=vz_group, defval='Body with wicks', title='Zones are cleare
int borderWidth = input.int(0, 'Zone border width', group=vz_group)
bool colorOverride = input.bool(true, 'Override color?' , group=vz_group, inline="vcz1")
color zoneColor = input.color(title='Color', group=vz_group, defval=color.rgb(255, 230, 75, 90), inlir
int transperancy = input.int(90, 'Zone Transperancy', minval = 0, maxval = 100, group=vz_group, toolti

bool overrideSym = input.bool(group='PVSRA Override', title='Override chart symbol?', defval=false, ir
string pvsraSym = input.string(group='PVSRA Override', title='', defval='INDEX:BTCUSD', tooltip='You c



pvsraVolume(overrideSymbolX, pvsraSymbolX, tickerIdX) =>
    request.security(overrideSymbolX ? pvsraSymbolX : tickerIdX, '', volume, barmerge.gaps_off, barmer
pvsraHigh(overrideSymbolX, pvsraSymbolX, tickerIdX) =>
    request.security(overrideSymbolX ? pvsraSymbolX : tickerIdX, '', high, barmerge.gaps_off, barmerge
pvsraLow(overrideSymbolX, pvsraSymbolX, tickerIdX) =>
    request.security(overrideSymbolX ? pvsraSymbolX : tickerIdX, '', low, barmerge.gaps_off, barmerge.
pvsraClose(overrideSymbolX, pvsraSymbolX, tickerIdX) =>
    request.security(overrideSymbolX ? pvsraSymbolX : tickerIdX, '', close, barmerge.gaps_off, barmerg
pvsraOpen(overrideSymbolX, pvsraSymbolX, tickerIdX) =>
    request.security(overrideSymbolX ? pvsraSymbolX : tickerIdX, '', open, barmerge.gaps_off, barmerge


pvsraVolume = pvsraVolume(overrideSym, pvsraSym, syminfo.tickerid)
pvsraHigh = pvsraHigh(overrideSym, pvsraSym, syminfo.tickerid)
```

```
pvsraLow = pvsraLow(overrideSym, pvsraSym, syminfo.tickerid)
pvsraClose = pvsraClose(overrideSym, pvsraSym, syminfo.tickerid)
pvsraOpen = pvsraOpen(overrideSym, pvsraSym, syminfo.tickerid)
[pvsraColor, alertFlag, averageVolume, volumeSpread, highestVolumeSpread] = trLib.calcPvsra(pvsraVolum


var zoneBoxesAbove = array.new_box()
var zoneBoxesBelow = array.new_box()


barcolor(setcandlecolors ? pvsraColor : na)
pvsra = trLib.getPvsraFlagByColor(switch_vectorzone ? pvsraColor:na, redVectorColor, greenVectorColor,
trLib.updateZones(pvsra, 0, zoneBoxesBelow, zonesMax, pvsraHigh, pvsraLow, pvsraOpen, pvsraClose, trar
trLib.updateZones(pvsra, 1, zoneBoxesAbove, zonesMax, pvsraHigh, pvsraLow, pvsraOpen, pvsraClose, trar
trLib.cleanarr(zoneBoxesAbove)
trLib.cleanarr(zoneBoxesBelow)
```

```
/*****************
// Market sessions
//*****************


string weekend_sessions = ':1234567'
string no_weekend_sessions = ':23456'

bool show_rectangle1 = input.bool(group='Market session: London (0800-1630 UTC+0) - DST Aware', defva
bool show_label1 = input.bool(group='Market session: London (0800-1630 UTC+0) - DST Aware', defval=tr
bool show_or1 = input.bool(group='Market session: London (0800-1630 UTC+0) - DST Aware', defval=true,
string sess1Label = input.string(group='Market session: London (0800-1630 UTC+0) - DST Aware', defval
color sess1col = input.color(group='Market session: London (0800-1630 UTC+0) - DST Aware', title='Col
color sess1colLabel = input.color(group='Market session: London (0800-1630 UTC+0) - DST Aware', title
string sess1TimeX = '0800-1630'//input.session(group='Market session: London (0800-1630 UTC+0)', defv
string rectStyle = input.string(group='Market session: London (0800-1630 UTC+0) - DST Aware', defval=
sessLineStyle = line.style_dashed
bool show_markets_weekends = input.bool(false, group='Market session: London (0800-1630 UTC+0) - DST


sess1Time = show_markets_weekends ? sess1TimeX + weekend_sessions : sess1TimeX + no_weekend_sessions


bool show_rectangle2 = input.bool(group='Market session: New York (1430-2100 UTC+0) - DST Aware', def
bool show_label2 = input.bool(group='Market session: New York (1430-2100 UTC+0) - DST Aware', defval=
bool show_or2 = input.bool(group='Market session: New York (1430-2100 UTC+0) - DST Aware', defval=tru
string sess2Label = input.string(group='Market session: New York (1430-2100 UTC+0) - DST Aware', defv
color sess2col = input.color(group='Market session: New York (1430-2100 UTC+0) - DST Aware', title='C
color sess2colLabel = input.color(group='Market session: New York (1430-2100 UTC+0) - DST Aware', tit
string sess2TimeX = '1430-2100'//input.session(group='Market session: New York (1430-2100 UTC+0)', de
sess2Time = show_markets_weekends ? sess2TimeX + weekend_sessions : sess2TimeX  + no_weekend_sessions


bool show_rectangle3 = input.bool(group='Market session: Tokyo (0000-0600 UTC+0) - DST Aware', defval
bool show_label3 = input.bool(group='Market session: Tokyo (0000-0600 UTC+0) - DST Aware', defval=tru
bool show_or3 = input.bool(group='Market session: Tokyo (0000-0600 UTC+0) - DST Aware', defval=true,
string sess3Label = input.string(group='Market session: Tokyo (0000-0600 UTC+0) - DST Aware', defval=
color sess3col = input.color(group='Market session: Tokyo (0000-0600 UTC+0) - DST Aware', title='Colo
color sess3colLabel = input.color(group='Market session: Tokyo (0000-0600 UTC+0) - DST Aware', title=
string sess3TimeX = '0000-0600'//input.session(group='Market session: Tokyo (0000-0600 UTC+0)', defva
sess3Time = show_markets_weekends ? sess3TimeX + weekend_sessions : sess3TimeX + no_weekend_sessions


bool show_rectangle4 = input.bool(group='Market session: Hong Kong (0130-0800 UTC+0) - DST Aware', de
bool show_label4 = input.bool(group='Market session: Hong Kong (0130-0800 UTC+0) - DST Aware', defval
bool show_or4 = input.bool(group='Market session: Hong Kong (0130-0800 UTC+0) - DST Aware', defval=tr
string sess4Label = input.string(group='Market session: Hong Kong (0130-0800 UTC+0) - DST Aware', def
color sess4col = input.color(group='Market session: Hong Kong (0130-0800 UTC+0) - DST Aware', title='
color sess4colLabel = input.color(group='Market session: Hong Kong (0130-0800 UTC+0) - DST Aware', ti
string sess4TimeX = '0130-0800'//input.session(group='Market session: Hong Kong (0130-0800 UTC+0)', d
sess4Time = show_markets_weekends ? sess4TimeX + weekend_sessions : sess4TimeX + no_weekend_sessions

bool show_rectangle5 = input.bool(group='Market session: Sydney (NZX+ASX 2200-0600 UTC+0) - DST Aware
bool show_label5 = input.bool(group='Market session: Sydney (NZX+ASX 2200-0600 UTC+0) - DST Aware', d
bool show_or5 = input.bool(group='Market session: Sydney (NZX+ASX 2200-0600 UTC+0) - DST Aware', defv
string sess5Label = input.string(group='Market session: Sydney (NZX+ASX 2200-0600 UTC+0) - DST Aware'
```

```
color sess5col = input.color(group='Market session: Sydney (NZX+ASX 2200-0600 UTC+0) - DST Aware', ti
color sess5colLabel = input.color(group='Market session: Sydney (NZX+ASX 2200-0600 UTC+0) - DST Aware
string sess5TimeX = '2200-0600'//input.session(group='Market session: Sydney (NZX+ASX 2200-0600 UTC+0
sess5Time = show_markets_weekends ? sess5TimeX + weekend_sessions : sess5TimeX + no_weekend_sessions

bool show_rectangle6 = input.bool(group='Market session: EU Brinks (0800-0900 UTC+0) - DST Aware', de
bool show_label6 = input.bool(group='Market session: EU Brinks (0800-0900 UTC+0) - DST Aware', defval
bool show_or6 = input.bool(group='Market session: EU Brinks (0800-0900 UTC+0) - DST Aware', defval=tr
string sess6Label = input.string(group='Market session: EU Brinks (0800-0900 UTC+0) - DST Aware', def
color sess6col = input.color(group='Market session: EU Brinks (0800-0900 UTC+0) - DST Aware', title='
color sess6colLabel = input.color(group='Market session: EU Brinks (0800-0900 UTC+0) - DST Aware', ti
string sess6TimeX = '0800-0900'//input.session(group='Market session: EU Brinks (0800-0900 UTC+0)', d
sess6Time = show_markets_weekends ? sess6TimeX + weekend_sessions : sess6TimeX + no_weekend_sessions

bool show_rectangle7 = input.bool(group='Market session: US Brinks (1400-1500 UTC+0) - DST Aware', de
bool show_label7 = input.bool(group='Market session: US Brinks (1400-1500 UTC+0) - DST Aware', defval
bool show_or7 = input.bool(group='Market session: US Brinks (1400-1500 UTC+0) - DST Aware', defval=tr
string sess7Label = input.string(group='Market session: US Brinks (1400-1500 UTC+0) - DST Aware', def
color sess7col = input.color(group='Market session: US Brinks (1400-1500 UTC+0) - DST Aware', title='
color sess7colLabel = input.color(group='Market session: US Brinks (1400-1500 UTC+0) - DST Aware', ti
string sess7TimeX = '1400-1500'//input.session(group='Market session: US Brinks (1400-1500 UTC+0)', d
sess7Time = show_markets_weekends ? sess7TimeX + weekend_sessions : sess7TimeX + no_weekend_sessions



splitSessionString(sessXTime) =>
    //session stirng looks like this: 0000-0000:1234567 ie start time, end time, day of the week
    //we need to parse the sessXTime string into hours and min for start and end times so we can use

    //string times contains "0000-2300" as an example
    string times = array.get(str.split(sessXTime, ':'), 0)

    //string startTime contains "0000"
    string startTime = array.get(str.split(times, '-'), 0)
    //string endTime contains "2300"
    string endTime = array.get(str.split(times, '-'), 1)

    //now we need to get the start hour and start min, sing 0 index - hour is the characters in index
    string[] startTimeChars = str.split(startTime, '')
    string[] endTimeChars = str.split(endTime, '')

    //so now startHour contains 00 and start min contains 00
    string startHour = array.get(startTimeChars, 0) + array.get(startTimeChars, 1)
    string startMin = array.get(startTimeChars, 2) + array.get(startTimeChars, 3)

    //so now endHour contains 23 and end min contains 00
    string endHour = array.get(endTimeChars, 0) + array.get(endTimeChars, 1)
    string endMin = array.get(endTimeChars, 2) + array.get(endTimeChars, 3)
    [startHour, startMin, endHour, endMin]

calc_session_startend(sessXTime, gmt) =>
    [startHour, startMin, endHour, endMin] = splitSessionString(sessXTime)
    targetstartTimeX = timestamp(gmt, year, month, dayofmonth, math.round(str.tonumber(startHour)), m
    targetendTimeX = timestamp(gmt, year, month, dayofmonth, math.round(str.tonumber(endHour)), math.
    time_now = timestamp(year, month, dayofmonth, hour, minute, 00)
    midnight_exchange = timestamp(year, month, dayofmonth, 00, 00, 00)
```

```
    midnight_exchange = timestamp(year, month, dayofmonth, 00, 00, 00)


    //if start hour is greater than end hour we are dealing with a session that starts towards the en
    //and ends the next day. ie advance the end time by 24 hours - its the next day
    bool adjusted = false
    if gmt == 'GMT+0'
        if math.round(str.tonumber(startHour)) > math.round(str.tonumber(endHour))
            if time_now - targetstartTimeX >= 0
                targetendTimeX := targetendTimeX + 24 * 60 * 60 * 1000
                adjusted := true
                targetendTimeX
    if gmt == 'GMT+1'
        if math.round(str.tonumber(startHour)) == 0
            startHour := '24'
        if math.round(str.tonumber(endHour)) == 0
            endHour := '24'
        if math.round(str.tonumber(startHour))-1 > math.round(str.tonumber(endHour))-1
            if time_now - targetstartTimeX >= 0
                targetendTimeX := targetendTimeX + 24 * 60 * 60 * 1000
                adjusted := true
                targetendTimeX



    if targetstartTimeX < midnight_exchange and midnight_exchange < targetendTimeX and not adjusted
        targetendTimeX := targetendTimeX + 24 * 60 * 60 * 1000
        targetendTimeX

    [targetstartTimeX,targetendTimeX]


draw_open_range(sessXTime, sessXcol, show_orX, gmt)=>
    if show_orX
        // Initialize variables on bar zero only, so they preserve their values across bars.
        var hi = float(na)
        var lo = float(na)
        var box hiLoBox = na
        // Detect changes in timeframe.
        session = time(timeframe.period, sessXTime, gmt)
        bool newTF = session and not session[1]
        if newTF
            // New bar in higher timeframe; reset values and create new lines and box.
            [targetstartTimeX,targetendTimeX] = calc_session_startend(sessXTime, gmt)
            sessionDuration = math.round(math.abs(time  - targetendTimeX)/(timeframe.multiplier*60*10

            hi := high
            lo := low
            hiLoBox := box.new(bar_index, hi, timeframe.multiplier == 1? bar_index : bar_index+sessic
            int(na)
        else
            if timeframe.multiplier == 1 and (na(session[1]) and not na(session) or session[1] < sess
                box.set_right(hiLoBox, bar_index+1)
            int(na)
draw_session_hilo(sessXTime, show_rectangleX, show_labelX, sessXcolLabel, sessXLabel, gmt)=>
    if show_rectangleX
        // Initialize variables on bar zero only, so they preserve their values across bars
```

```
        // Initialize variables on bar zero only, so they preserve their values across bars.
        var hi = float(0)
        var lo = float(10000000000.0)

        var line line_t = na
        var line line_b = na
        var label line_label = na
        // var box hiLoBox = na
        // Detect changes in timeframe.
        session = time(timeframe.period, sessXTime, gmt)
        sessLineStyleX = rectStyle == 'Solid' ? line.style_solid : line.style_dashed
        bool newTF = session and not session[1]
        hi := newTF ? high : session ? math.max(high, hi[1]) : hi[1]
        lo := newTF ? low : session ? math.min(low, lo[1]) : lo[1]

        if newTF
            beginIndex = bar_index
            [targetstartTimeX,targetendTimeX] = calc_session_startend(sessXTime, gmt)
            sessionDuration = math.round(math.abs(time  - targetendTimeX)/(timeframe.multiplier*60*10

            line_t := line.new(beginIndex, hi, timeframe.multiplier == 1? bar_index : bar_index+sessi
            line_b := line.new(beginIndex, lo, timeframe.multiplier == 1? bar_index : bar_index+sessi
            line.delete(line_t[1])
            line.delete(line_b[1])
            if show_labelX
                line_label := label.new(beginIndex, hi, sessXLabel, xloc=xloc.bar_index, textcolor=se
                label.delete(line_label[1])

            int(na)
        else
            if na(session[1]) and not na(session) or session[1] < session
                if timeframe.multiplier == 1
                    line.set_x2(line_t,bar_index+1)
                    line.set_x2(line_b,bar_index+1)
                line.set_y1(line_t,hi)
                line.set_y2(line_t,hi)
                line.set_y1(line_b,lo)
                line.set_y2(line_b,lo)
                if show_labelX and not na(line_label)
                    label.set_y(line_label, hi)
            int(na)




///*****************************/
// Daylight Savings Time Flags //
//*****************************//

int previousSunday = dayofmonth - dayofweek + 1
bool nyDST = na
bool ukDST = na
bool sydDST = na

if month < 3 or month > 11
    nyDST := false
```

```
    ukDST := false
    sydDST := true
else if month > 4 and month < 10
    nyDST := true
    ukDST := true
    sydDST := false
else if month == 3
    nyDST := previousSunday >= 8
    ukDST := previousSunday >= 24
    sydDST := true
else if month == 4
    nyDST := true
    ukDST := true
    sydDST := previousSunday <= 0
else if month == 10
    nyDST := true
    ukDST := previousSunday <= 24
    sydDST := previousSunday >= 0
else // month == 11
    nyDST := previousSunday <= 0
    ukDST := false
    sydDST := true




if ukDST
    draw_open_range(sess1Time,sess1col,show_or1,'GMT+1')
    draw_session_hilo(sess1Time, show_rectangle1, show_label1, sess1colLabel, sess1Label, 'GMT+1')
else
    draw_open_range(sess1Time,sess1col,show_or1,'GMT+0')
    draw_session_hilo(sess1Time, show_rectangle1, show_label1, sess1colLabel, sess1Label, 'GMT+0')

if nyDST
    draw_open_range(sess2Time,sess2col,show_or2,'GMT+1')
    draw_session_hilo(sess2Time, show_rectangle2, show_label2, sess2colLabel, sess2Label, 'GMT+1')
else
    draw_open_range(sess2Time,sess2col,show_or2,'GMT+0')
    draw_session_hilo(sess2Time, show_rectangle2, show_label2, sess2colLabel, sess2Label, 'GMT+0')

// Tokyo
draw_open_range(sess3Time,sess3col,show_or3,'GMT+0')
draw_session_hilo(sess3Time, show_rectangle3, show_label3, sess3colLabel, sess3Label, 'GMT+0')

// Hong Kong
draw_open_range(sess4Time,sess4col,show_or4,'GMT+0')
draw_session_hilo(sess4Time, show_rectangle4, show_label4, sess4colLabel, sess4Label, 'GMT+0')

if sydDST
    draw_open_range(sess5Time,sess5col,show_or5,'GMT+1')
    draw_session_hilo(sess5Time, show_rectangle5, show_label5, sess5colLabel, sess5Label, 'GMT+1')
else
    draw_open_range(sess5Time,sess5col,show_or5,'GMT+0')
    draw_session_hilo(sess5Time, show_rectangle5, show_label5, sess5colLabel, sess5Label, 'GMT+0')
```

```
if nyDST
    draw_open_range(sess7Time,sess7col,show_or7,'GMT+1')
    draw_session_hilo(sess7Time, show_rectangle7, show_label7, sess7colLabel, sess7Label, 'GMT+1')
else
    draw_open_range(sess7Time,sess7col,show_or7,'GMT+0')
    draw_session_hilo(sess7Time, show_rectangle7, show_label7, sess7colLabel, sess7Label, 'GMT+0')
```

```
////////////////////////////////////////////////
////QQE MOD
////////////////////////////////////////////////

qqe_gorup = "█████████ QQE █████████"
RSI_Period = input(6, title='RSI Length', group=qqe_gorup)
SF = input(5, title='RSI Smoothing', group=qqe_gorup)
QQE = input(3, title='Fast QQE Factor', group=qqe_gorup)
ThreshHold = input(3, title='Thresh-hold', group=qqe_gorup)
//

srcqqe = input(close, title='RSI Source', group=qqe_gorup)
//

//
Wilders_Period = RSI_Period * 2 - 1


Rsi = ta.rsi(srcqqe, RSI_Period)
RsiMa = ta.ema(Rsi, SF)
AtrRsi = math.abs(RsiMa[1] - RsiMa)
MaAtrRsi = ta.ema(AtrRsi, Wilders_Period)
dar = ta.ema(MaAtrRsi, Wilders_Period) * QQE

longband = 0.0
shortband = 0.0
trend = 0

DeltaFastAtrRsi = dar
RSIndex = RsiMa
newshortband = RSIndex + DeltaFastAtrRsi
newlongband = RSIndex - DeltaFastAtrRsi
longband := RSIndex[1] > longband[1] and RSIndex > longband[1] ? math.max(longband[1], newlongband) :
shortband := RSIndex[1] < shortband[1] and RSIndex < shortband[1] ? math.min(shortband[1], newshortbar
cross_1 = ta.cross(longband[1], RSIndex)
trend := ta.cross(RSIndex, shortband[1]) ? 1 : cross_1 ? -1 : nz(trend[1], 1)
FastAtrRsiTL = trend == 1 ? longband : shortband


length = input.int(50, minval=1, title='Bollinger Length', group=qqe_gorup)
multqqe = input.float(0.35, minval=0.001, maxval=5, step=0.1, title='BB Multiplier', group=qqe_gorup)
basis = ta.sma(FastAtrRsiTL - 50, length)
dev = multqqe * ta.stdev(FastAtrRsiTL - 50, length)
```

```
upperqqe = basis + dev
lowerqqe = basis - dev
color_bar = RsiMa - 50 > upperqqe ? #00c3ff : RsiMa - 50 < lowerqqe ? #ff0062 : color.gray


//
// Zero cross
QQEzlong = 0
QQEzlong := nz(QQEzlong[1])
QQEzshort = 0
QQEzshort := nz(QQEzshort[1])
QQEzlong := RSIndex >= 50 ? QQEzlong + 1 : 0
QQEzshort := RSIndex < 50 ? QQEzshort + 1 : 0
//


/////////////////////////////////////////////////////////////

RSI_Period2 = input(6, title='RSI Length', group=qqe_gorup)
SF2 = input(5, title='RSI Smoothing', group=qqe_gorup)
QQE2 = input(1.61, title='Fast QQE2 Factor', group=qqe_gorup)
ThreshHold2 = input(3, title='Thresh-hold', group=qqe_gorup)

src2 = input(close, title='RSI Source', group=qqe_gorup)
//

//
Wilders_Period2 = RSI_Period2 * 2 - 1


Rsi2 = ta.rsi(src2, RSI_Period2)
RsiMa2 = ta.ema(Rsi2, SF2)
AtrRsi2 = math.abs(RsiMa2[1] - RsiMa2)
MaAtrRsi2 = ta.ema(AtrRsi2, Wilders_Period2)
dar2 = ta.ema(MaAtrRsi2, Wilders_Period2) * QQE2
longband2 = 0.0
shortband2 = 0.0
trend2 = 0

DeltaFastAtrRsi2 = dar2
RSIndex2 = RsiMa2
newshortband2 = RSIndex2 + DeltaFastAtrRsi2
newlongband2 = RSIndex2 - DeltaFastAtrRsi2
longband2 := RSIndex2[1] > longband2[1] and RSIndex2 > longband2[1] ? math.max(longband2[1], newlongba
shortband2 := RSIndex2[1] < shortband2[1] and RSIndex2 < shortband2[1] ? math.min(shortband2[1], newsh
cross_2 = ta.cross(longband2[1], RSIndex2)
trend2 := ta.cross(RSIndex2, shortband2[1]) ? 1 : cross_2 ? -1 : nz(trend2[1], 1)
FastAtrRsi2TL = trend2 == 1 ? longband2 : shortband2


//
// Zero cross
QQE2zlong = 0
QQE2zlong := nz(QQE2zlong[1])
QQE2zshort = 0
```

```
QQE2zshort := nz(QQE2zshort[1])
QQE2zlong := RSIndex2 >= 50 ? QQE2zlong + 1 : 0
QQE2zshort := RSIndex2 < 50 ? QQE2zshort + 1 : 0
//
qqeline = FastAtrRsi2TL - 50
// hcolor2 = RsiMa2 - 50 > ThreshHold2 ? color.silver : RsiMa2 - 50 < 0 - ThreshHold2 ? color.silver :


Greenbar1 = RsiMa2 - 50 > ThreshHold2
Greenbar2 = RsiMa - 50 > upperqqe

Redbar1 = RsiMa2 - 50 < 0 - ThreshHold2
Redbar2 = RsiMa - 50 < lowerqqe



////////////////////////////////////////////////
/////////////// Volume Up/Down
////////////////////////////////////////////////
vgroup = "██████████ Up/Down Volume ██████████"
lowerTimeframeTooltip = "The indicator scans lower timeframe data to approximate Up/Down volume. By de
useCustomTimeframeInput = input.bool(false, "Use custom timeframe", tooltip = lowerTimeframeTooltip,gr
lowerTimeframeInput = input.timeframe("1", "Timeframe",group=vgroup)

upAndDownVolume() =>
    posVol = 0.0
    negVol = 0.0

    switch
        close >  open     => posVol += volume
        close <  open     => negVol -= volume
        close >= close[1] => posVol += volume
        close <  close[1] => negVol -= volume

    [posVol, negVol]

lowerTimeframe = switch
    useCustomTimeframeInput => lowerTimeframeInput
    timeframe.isintraday    => "1"
    timeframe.isdaily       => "5"
    => "60"

// Modify the timeframe argument in the security call
timeframeForSecurity = useCustomTimeframeInput ? lowerTimeframeInput : timeframe.period

[upVolumeArray, downVolumeArray] = request.security_lower_tf(syminfo.tickerid, timeframeForSecurity, u



SMALength = input(20, title="SMA Length",group=vgroup)
volume_ma = ta.sma(volume, SMALength)

volume_abovema_signal = volume > volume_ma

upVolume = array.sum(upVolumeArray)
```

```
downVolume = array.sum(downVolumeArray)
delta = upVolume + downVolume
prevdelta = delta[1]


var cumVol = 0.
cumVol += nz(volume)
if barstate.islast and cumVol == 0
    runtime.error("The data vendor doesn't provide volume data for this symbol.")




respectemavalue = ta.ema(src, respectemaperiod)
isaboverespectema = close > respectemavalue
isbelowrespectema = close < respectemavalue



isqqebarabove = Greenbar1 and Greenbar2
isqqebarbelow = Redbar1 and Redbar2



dv2up = bool (na)



dvup = bool (na)

if dvtype == 'Threshold'
    dvup := vol > t and vol>=1.1
else if dvtype == '10p Difference'
    dvup := vol > t and (vol - t >= 0.1)
else
    dvup := vol > t


sarup = out < close
sardown = out > close

longvol = bool(na)
shortvol = bool (na)



if volumetype == 'Delta'
    longvol := delta > 0 and delta > delta[1]
    shortvol := delta < 0 and delta < delta[1]
else if volumetype == 'volume above MA'
    longvol := volume_abovema_signal
    shortvol := volume_abovema_signal
else
    longvol := upVolume > upVolume[1]
    shortvol := downVolume < downVolume[1]
```

```
longCond = bool(na)
shortCond = bool(na)


longCond2 = bool(na)
shortCond2 = bool(na)

vipcondition = bool(na)
vimcondition = bool(na)


if vitype == 'Simple'
    vipcondition := vip > vim
    vimcondition := vip < vim
else
    vipcondition := vip > vim and vip > viupper and vip > vip[1] and vim < vim[1] and vim[1] <= vilowe
    vimcondition := vip < vim and vim > viupper and vim > vim[1] and vip < vip [1] and vip[1] <= vilow




vipcondition2 = vip > vim
vimcondition2 = vip < vim

////////////////////////////////////ADX Condition ////////////////////////////////////////
adxcycle = 0
adxup = ta.crossover(adx, keyLevel)
adxdown = ta.crossunder(adx, keyLevel)
adxcycle := adxup ? 1 : adxdown ? -1 : adxcycle[1]
adxcondition = string(na)

adxupcondition = bool(na)
adxdowncondition = bool (na)
if adxtype == 'Adx & +Di -Di'
    adxupcondition := diplus > diminus and adx>=keyLevel
    adxdowncondition := diplus < diminus and adx>=keyLevel
if adxtype == 'Adx Only'
    adxupcondition := adx>keyLevel
    adxdowncondition := adx>keyLevel
else
    if adxcycle == -1
        adxupcondition := diplus > diminus and adx>=keyLevel and diplus - diminus > 1
        adxdowncondition := diplus < diminus and adx>=keyLevel and diminus - diplus > 1
    else if adxcycle==1
        adxupcondition := diplus > diminus and adx>=keyLevel and adx<55 and (adx>adx[1] or (diplus > d
        adxdowncondition := diplus < diminus and adx>=keyLevel and adx<55 and (adx>adx[1] or (diplus <




///////////////adx condition end//////////////////////////////////////////////////////
```

```
justcontinue = bool(true)

isstup = bool(na)
isstdown = bool(na)
isstup := sttrend == 1
isstdown := sttrend != 1

ismacdup = bool(na)
ismacddown = bool(na)

isqqeabove = bool(na)
isqqebelow = bool(na)

if qqetype == 'Line'
    isqqeabove := qqeline>0
    isqqebelow := qqeline<0
else if qqetype == 'Bar'
    isqqeabove := RsiMa2 - 50 > 0 and (Greenbar1 and Greenbar2)
    isqqebelow := RsiMa2 - 50 < 0 and (Redbar1 and Redbar2)
else if qqetype == 'Line & Bar'
    isqqeabove := RsiMa2 - 50 > 0 and (Greenbar1 and Greenbar2) and qqeline>0
    isqqebelow := RsiMa2 - 50 < 0 and (Redbar1 and Redbar2) and qqeline<0



rsimalong2 = bool(na)
rsimashort2 = bool(na)

rsimalong2:= rsiMA >= rsiMA[1]
rsimashort2:= rsiMA <= rsiMA[1]


rsilimitlong = rsi >= rsilimitup
rsilimitshort = rsi <= rsilimitdown

rsimalimitlong = rsiMA >= rsimalimitup
rsimalimitshort = rsiMA <= rsimalimitdown



leadinglongcond = bool(na)
leadingshortcond = bool(na)


if leadingindicator == 'Range Filter'

    if rftype == 'Default'
        leadinglongcond := src > filt and src > src[1] and upward > 0 or src > filt and src < src[1] a
        leadingshortcond := src < filt and src < src[1] and downward > 0 or src < filt and src > src[1
    else if rftype == 'DW'
        leadinglongcond := rfupward
        leadingshortcond := rfdownward

else if leadingindicator == 'DMI (Adx)'
```

```
    if adxtype == 'Basic'
        leadinglongcond := diplus > diminus and adx>=keyLevel
        leadingshortcond := diplus < diminus and adx>=keyLevel
    else
        if adxcycle == -1
            leadinglongcond := diplus > diminus and adx>=keyLevel and diplus - diminus > 1
            leadingshortcond := diplus < diminus and adx>=keyLevel and diminus - diplus > 1
        else if adxcycle==1
            leadinglongcond := diplus > diminus and adx>=keyLevel and adx<55 and (adx>adx[1] or (diplu
            leadingshortcond := diplus < diminus and adx>=keyLevel and adx<55 and (adx>adx[1] or (dipl
else if leadingindicator == 'Parabolic SAR (PSAR)'
    leadinglongcond := out < close
    leadingshortcond := out > close
else if leadingindicator == 'Rational Quadratic Kernel (RQK)'
    leadinglongcond := rqkuptrend
    leadingshortcond := rqkdowntrend

else if leadingindicator == 'Trendline Breakout'
    leadinglongcond := tb_buysignal
    leadingshortcond := tb_sellsignal


else if leadingindicator == 'Range Detector'
    leadinglongcond := rd_long
    leadingshortcond := rd_short


else if leadingindicator == 'Heiken-Ashi Candlestick Oscillator'
    leadinglongcond := hacolt_long
    leadingshortcond := hacolt_short



else if leadingindicator == 'Donchian Trend Ribbon'
    leadinglongcond := donchian_long
    leadingshortcond := donchian_short

else if leadingindicator == 'Rate of Change (ROC)'
    leadinglongcond := roc_long
    leadingshortcond := roc_short

else if leadingindicator == 'Trend Direction Force Index (TDFI)'
    leadinglongcond := tdfi_long
    leadingshortcond := tdfi_short




else if leadingindicator == 'Detrended Price Oscillator (DPO)'
    leadinglongcond := dpo_long
    leadingshortcond := dpo_short

else if leadingindicator == '2 EMA Cross'
    leadinglongcond := first_2ema > second_2ema
    leadingshortcond := first_2ema < second_2ema
```

```
else if leadingindicator == '3 EMA Cross'
    leadinglongcond := first_3ema > second_3ema and first_3ema > third_3ema and second_3ema>third_3ema
    leadingshortcond := first_3ema < second_3ema and first_3ema < third_3ema and second_3ema<third_3em

else if leadingindicator == 'Chandelier Exit'
    leadinglongcond := ce_long
    leadingshortcond := ce_short

else if leadingindicator == 'Stochastic'
    leadinglongcond := stoch_long
    leadingshortcond := stoch_short

else if leadingindicator == 'Vortex Index'

    if vitype == 'Simple'
        leadinglongcond := vip > vim
        leadingshortcond := vip < vim
    else
        leadinglongcond := vip > vim and vip > viupper and vip > vip[1] and vim < vim[1] and vim[1] <=
        leadingshortcond := vip < vim and vim > viupper and vim > vim[1] and vip < vip [1] and vip[1]

else if leadingindicator == 'Schaff Trend Cycle (STC)'
    leadinglongcond := stc >= upper
    leadingshortcond := stc <= upper
else if leadingindicator == 'Wolfpack Id'
    leadinglongcond := wolf_long
    leadingshortcond := wolf_short


else if leadingindicator == 'B-Xtrender'
    leadinglongcond := bx_long
    leadingshortcond := bx_short


else if leadingindicator == 'Bull Bear Power Trend'
    leadinglongcond := bbpt_long
    leadingshortcond := bbpt_short

else if leadingindicator == 'QQE Mod'
    if qqetype == 'Line'
        leadinglongcond := qqeline>0
        leadingshortcond := qqeline<0
    else if qqetype == 'Bar'
        leadinglongcond := RsiMa2 - 50 > 0 and (Greenbar1 and Greenbar2)
        leadingshortcond := RsiMa2 - 50 < 0 and (Redbar1 and Redbar2)
    else if qqetype == 'Line & Bar'
        leadinglongcond := RsiMa2 - 50 > 0 and (Greenbar1 and Greenbar2) and qqeline>0
        leadingshortcond := RsiMa2 - 50 < 0 and (Redbar1 and Redbar2) and qqeline<0



else if leadingindicator == 'MACD'
    if macdtype == 'MACD Crossover'
```

```
        leadinglongcond := macdd > signal
        leadingshortcond := macdd < signal
    else if macdtype == 'Zero line crossover'
        leadinglongcond := macdd > signal and macdd > 0.00000
        leadingshortcond := macdd < signal and macdd < 0.00000


else if leadingindicator == 'True Strength Indicator (TSI)'
    if tsitype == 'Signal Cross'
        leadinglongcond := tsi_long
        leadingshortcond := tsi_short
    else if tsitype == 'Zero line cross'
        leadinglongcond := tsi_long
        leadingshortcond := tsi_short


else if leadingindicator == 'RSI'
    if rsitype == 'RSI MA Cross'
        leadinglongcond := rsi > rsiMA
        leadingshortcond := rsi < rsiMA
    else if rsitype == 'RSI Exits OB/OS zones'
        leadinglongcond := rsi > rsi_lower and rsi[1] < rsi_lower
        leadingshortcond := rsi < rsi_upper and rsi[1] > rsi_upper
    else if rsitype == 'RSI Level'
        leadinglongcond := rsi > respectrsilevel
        leadingshortcond := rsi < respectrsilevel

else if leadingindicator == 'Chaikin Money Flow'
    leadinglongcond := chaikin_long
    leadingshortcond := chaikin_short



else if leadingindicator == 'Volatility Oscillator'
    leadinglongcond := vo_long
    leadingshortcond := vo_short


else if leadingindicator == 'SSL Channel'
    leadinglongcond := ssl_long
    leadingshortcond := ssl_short

else if leadingindicator == 'Awesome Oscillator'
    leadinglongcond := ao_long
    leadingshortcond := ao_short


else if leadingindicator == 'Supertrend'

    leadinglongcond := sttrend == 1
    leadingshortcond := sttrend != 1

else if leadingindicator == 'Half Trend'

    leadinglongcond := halftrend_long
```

```
    leadingshortcond := halftrend_short


else if leadingindicator == 'Waddah Attar Explosion'

    leadinglongcond := wae_long
    leadingshortcond := wae_short


else if leadingindicator == 'Hull Suite'
    leadinglongcond := HULL > HULL[2]
    leadingshortcond := HULL < HULL[2]


else if leadingindicator == 'BB Oscillator'
    leadinglongcond := bbosc_long
    leadingshortcond := bbosc_short

else if leadingindicator == 'Ichimoku Cloud'
    leadinglongcond := ichi_long
    leadingshortcond := ichi_short

else if leadingindicator == 'VWAP'
    leadinglongcond := long_vwap
    leadingshortcond := short_vwap

else if leadingindicator == 'SuperIchi'
    leadinglongcond := superichi_long
    leadingshortcond := superichi_short

else if leadingindicator == 'Trend Meter'
    if tmtype == '3 TM and 2 TB change to same color'
        leadinglongcond := TB1Green and TB2Green  and (TrendBar1Result and TrendBar2Result and TrendBa
        leadingshortcond := TB1Red and TB2Red and (not TrendBar1Result and not TrendBar2Result and not
    else if tmtype == '3 TM change to same color'
        leadinglongcond := TrendBar1Result and TrendBar2Result and TrendBar3Result ? 1 : 0
        leadingshortcond := not TrendBar1Result and not TrendBar2Result and not TrendBar3Result ? 1 :
    else if tmtype == '3 TM, 2 TB and Wavetrend change to same color'
        leadinglongcond := TB1Green and TB2Green  and (TrendBar1Result and TrendBar2Result and TrendBa
        leadingshortcond := TB1Red and TB2Red and (not TrendBar1Result and not TrendBar2Result and not
else if leadingindicator == 'CCI'
    leadinglongcond := ccilong
    leadingshortcond := ccishort


tmup = bool(na)
tmdown = bool(na)

if tmtype == '3 TM and 2 TB change to same color'
    tmup := TB1Green and TB2Green  and (TrendBar1Result and TrendBar2Result and TrendBar3Result ? 1 :
    tmdown := TB1Red and TB2Red and (not TrendBar1Result and not TrendBar2Result and not TrendBar3Resu
else if tmtype == '3 TM change to same color'
    tmup := TrendBar1Result and TrendBar2Result and TrendBar3Result ? 1 : 0
    tmdown := not TrendBar1Result and not TrendBar2Result and not TrendBar3Result ? 1 : 0
else if tmtype == '3 TM, 2 TB and Wavetrend change to same color'
```

```
    tmup := TB1Green and TB2Green  and (TrendBar1Result and TrendBar2Result and TrendBar3Result ? 1 :
    tmdown := TB1Red and TB2Red and (not TrendBar1Result and not TrendBar2Result and not TrendBar3Resu

hullup = bool(na)
hulldown = bool(na)
if respecthull
    hullup := HULL > HULL[2]
    hulldown := HULL < HULL[2]

rsiup = bool (na)
rsidown = bool (na)

if rsitype == 'RSI MA Cross'
    rsiup := rsi > rsiMA
    rsidown := rsi < rsiMA
else if rsitype == 'RSI Exits OB/OS zones'
    rsiup := rsi > rsi_lower and rsi[1] < rsi_lower
    rsidown := rsi < rsi_upper and rsi[1] > rsi_upper
else if rsitype == 'RSI Level'
    rsiup := rsi > respectrsilevel
    rsidown := rsi < respectrsilevel


if macdtype == 'MACD Crossover'
    ismacdup := macdd > signal
    ismacddown := macdd < signal
else if macdtype == 'Zero line crossover'
    ismacdup := macdd > signal and macdd > 0.00000
    ismacddown := macdd < signal and macdd < 0.00000

ema2_long = first_2ema > second_2ema
ema2_short = first_2ema < second_2ema

uprf = bool (na)
downrf = bool(na)


if rftype == 'Default'
    uprf := src > filt and src > src[1] and upward > 0 or src > filt and src < src[1] and upward > 0
    downrf := src < filt and src < src[1] and downward > 0 or src < filt and src > src[1] and downwarc
else if rftype == 'DW'
    uprf := rfupward
    downrf := rfdownward

ema3_long = first_3ema > second_3ema and first_3ema > third_3ema and second_3ema>third_3ema
ema3_short = first_3ema < second_3ema and first_3ema < third_3ema and second_3ema<third_3ema


longCond := leadinglongcond  and   (respectrf?uprf:justcontinue) and
 (respectadx?adxupcondition:justcontinue)  and (respecttsi?tsi_long:justcontinue) and (respecthacolt?h
  (respectvol?longvol:justcontinue) and (respectchaikin?chaikin_long:justcontinue)  and (respectvwap?l

shortCond := leadingshortcond and   (respectrf?downrf:justcontinue)  and
 (respectadx?adxdowncondition:justcontinue) and (respecttsi?tsi_short:justcontinue)   and (respecthacc
  (respectvol?shortvol:justcontinue) and (respectvwap?short_vwap:justcontinue)  and (respectvo?vo_shor
```

```
var int leadinglong_count = 0
var int leadinglong_count2 = 0

var int leadingshort_count = 0
var int leadingshort_count2 = 0


if leadinglongcond
    leadinglong_count := leadinglong_count + 1
    leadinglong_count2 := leadinglong_count

for i = 1 to 100
    if leadinglongcond[i]
        leadinglong_count := leadinglong_count + 1
        leadinglong_count2 := leadinglong_count

    else
        leadinglong_count := 0
        break


if leadingshortcond
    leadingshort_count := leadingshort_count + 1
    leadingshort_count2 := leadingshort_count

for i = 1 to 100
    if leadingshortcond[i]
        leadingshort_count := leadingshort_count + 1
        leadingshort_count2 := leadingshort_count

    else
        leadingshort_count := 0
        break

CondIni = 0

longCondition = bool (na)
shortCondition = bool(na)




// if expiry option is used
longcond_withexpiry = longCond and leadinglong_count2 <= signalexpiry
shortcond_withexpiry = shortCond and leadingshort_count2 <= signalexpiry

log.info("leadinglong_count2 : {0}",leadinglong_count2)
log.info("leadingshort_count2: {0}",leadingshort_count2)
//without expiry
longCondition := longcond_withexpiry and CondIni[1] == -1
```

```
shortCondition := shortcond_withexpiry and CondIni[1] == 1



if alternatesignal
    longCondition := longcond_withexpiry and CondIni[1] == -1
    shortCondition := shortcond_withexpiry and CondIni[1] == 1

else
    longCondition := longcond_withexpiry
    shortCondition := shortcond_withexpiry



CondIni := longcond_withexpiry ? 1 : shortcond_withexpiry ? -1 : CondIni[1]


is_expiry_count_crossed_long = leadinglong_count2 >= signalexpiry
is_expiry_count_crossed_short = leadingshort_count2 >= signalexpiry




plotshape(showsignal ? (longCondition[1] ? false : longCondition) : na, title='Buy Signal', text='long
plotshape(showsignal and shortCondition and showsignal, title='Sell Signal', text='short', textcolor=c


alertcondition(longCondition, title='Buy Alert', message='BUY')
alertcondition(shortCondition, title='Sell Alert', message='SELL')



rsitype2 = rsitype
if rsitype2 == "RSI Level"
    rsitype2 := "RSI Level (" + str.tostring(respectrsilevel) +")"


confirmation_counter = array.new_string(0)
confirmation_val = array.new_string(0)
confirmation_val_short = array.new_string(0)




pushConfirmation(respect, label, longCondition, shortCondition) =>
    if respect
        array.push(confirmation_counter, label)
        array.push(confirmation_val, longCondition ? "✔" : "✘")
        array.push(confirmation_val_short, shortCondition ? "✔" : "✘")

pushConfirmation(respectema, "EMA", isaboverespectema, isbelowrespectema)
pushConfirmation(respect2ma, "2 EMA Cross (" + str.tostring(respect2maperiod_1) + "," + str.tostring(r
pushConfirmation(respect3ma, "3 EMA Cross (" + str.tostring(respect3maperiod_1) + "," + str.tostring(r
```

```
pushConfirmation(respectrf, "Range Filter", uprf, downrf)
pushConfirmation(respectrqk, "Rational Quadratic Kernel (RQK)", rqkuptrend, rqkdowntrend)
pushConfirmation(respectst, "SuperTrend", isstup, isstdown)
pushConfirmation(respectht, "Half Trend", halftrend_long, halftrend_short)

pushConfirmation(respecttrendline_breakout, "Trendline breakout", tb_buysignal, tb_sellsignal)
pushConfirmation(respectrd, "Range Detector", rd_long, rd_short)
pushConfirmation(respecthacolt, "Heiken-Ashi Candlestick Oscillator", hacolt_long, hacolt_short)


pushConfirmation(respectdonchian, "Donchian Trend Ribbon", donchian_long, donchian_short)
pushConfirmation(respectroc, "Rate of Change (ROC)", roc_long, roc_short)
pushConfirmation(respecttsi, "True Strength Indicator (TSI)", tsi_long, tsi_short)

pushConfirmation(respecttdfi, "Trend Direction Force Index (TDFI)", tdfi_long, tdfi_short)


pushConfirmation(respectbx, "B-Xtrender (" + str.tostring(bxtype) + ")", bx_long, bx_short)
pushConfirmation(respectbbpt, "Bull Bear Power Trend (" + str.tostring(bbpttype) + ")", bbpt_long, bbp
pushConfirmation(respectvwap, "VWAP", long_vwap, short_vwap)
pushConfirmation(respectichi, "Ichimoku Cloud", ichi_long, ichi_short)
pushConfirmation(respectsuperichi, "Superichi", superichi_long, superichi_short)
pushConfirmation(respectbbosc, "BB Oscillator", bbosc_long, bbosc_short)
pushConfirmation(respecttm, "Trend Meter", tmup, tmdown)
pushConfirmation(respectce, "Chandelier Exit", ce_long, ce_short)
pushConfirmation(respectcci, "CCI", ccilong, ccishort)
pushConfirmation(respectadx, "DMI (Adx)  (" + str.tostring(adxtype) + ")", adxupcondition, adxdownconc
pushConfirmation(respectsar, "Parabolic SAR", sarup, sardown)
pushConfirmation(respectssl, "SSL Channel", ssl_long, ssl_short)
pushConfirmation(respectvo, "Volatility Oscillator", vo_long, vo_short)
pushConfirmation(respectdpo, "Detrended Price Oscillator(DPO)", dpo_long, dpo_short)
pushConfirmation(respectmd, "McGinley Dynamic", md_long, md_short)

pushConfirmation(respectdv, "DV", dvup, dvup) // Note: Both are 'dvup'. Double-check if it's intended.
pushConfirmation(respectci, "Choppiness Index", ci_filter, ci_filter) // Note: Both are 'dvup'. Double

pushConfirmation(respectstochastic, "Stochastic (" + str.tostring(stochtype) + ")", stoch_long, stoch_
pushConfirmation(respectrsi, "RSI (" + str.tostring(rsitype2) + ")", rsiup, rsidown)
pushConfirmation(respectmacd, "MACD (" + str.tostring(macdtype) + ")", ismacdup, ismacddown)
pushConfirmation(respectstc, "Schaff Trend Cycle", stcup, stcdown)
pushConfirmation(respectwae, "Waddah Attar Explosion", wae_long, wae_short)

pushConfirmation(respectchaikin, "Chaikin Money Flow", chaikin_long, chaikin_short)
pushConfirmation(respectvol, "Volume", longvol, shortvol)
pushConfirmation(respectao, "Awesome Oscillator(" + str.tostring(aotype) + ")", ao_long, ao_short)
pushConfirmation(respectwolf, "Wolfpack Id", wolf_long, wolf_short)
pushConfirmation(respectqqe, "QQE Mod (" + str.tostring(qqetype) + ")", isqqeabove, isqqebelow)
pushConfirmation(respecthull, "HullSuite", hullup, hulldown)
pushConfirmation(respectvi, "Vortex Index  (" + str.tostring(vitype) + ")", vipcondition, vimconditior



getFalseShortConditionItems(arrShort) =>
    sShort = ""
    if array.size(arrShort) > 0
```

```
        for i = 0 to array.size(arrShort) - 1
            if array.get(arrShort, i) == "❌"
                sShort := sShort + array.get(confirmation_counter, i) + "\n"
    sShort

falseShortConditionString = getFalseShortConditionItems(confirmation_val_short)

tooltipS = "🔴 Failed Short Confirmations: " + "\n" + "-------------------------" + "\n" + falseShor

labelShortCondition = showsignal and is_expiry_count_crossed_short and not is_expiry_count_crossed_shc

if labelShortCondition
    label.new(bar_index, low, "🔴", color=color.new(#dd1111, 0), textcolor=color.white, style=label.s



getFalseLongConditionItems(arr) =>
    s = ""
    if array.size(arr) > 0
        for i = 0 to array.size(arr) - 1
            if array.get(arr, i) == "❌"
                s := s + array.get(confirmation_counter, i) + "\n"
    s

falseLongConditionString = getFalseLongConditionItems(confirmation_val)

tooltipL = "🔴 Failed Long Confirmations: " + "\n" + "-------------------------" + "\n" + falseLongC

labelCondition = showsignal and is_expiry_count_crossed_long and not is_expiry_count_crossed_long[1] a

if labelCondition
    label.new(bar_index, high, "🔴", color=color.new(color.green, 0), textcolor=color.white, style=la







leadingstatus = leadinglongcond ? "✔" : "❌"
leadingstatus_short = leadingshortcond ? "✔" : "❌"


rowcount = int(na)
if array.size(confirmation_counter) ==0
    rowcount := 5
else
    rowcount := array.size(confirmation_counter)+4


if showdashboard
    var table tab1 = table.new(i_tab1Ypos + '_' + i_tab1Xpos, 3, rowcount, color.rgb(42, 46, 57), colc


    table.cell(tab1, 1, 0,"Long", text_halign=text.align_left, text_size=table_size(in_dashboardtab_si
    table.cell(tab1, 2, 0,"Short", text_halign=text.align_left, text_size=table_size(in_dashboardtab_s
```

```
        table.cell(tab1, 0, 1, "Leading Indicator", text_halign=text.align_left, text_size=table_size(in_d
        table.cell(tab1, 1, 1,"", text_halign=text.align_left, text_size=table_size(in_dashboardtab_size),
        table.cell(tab1, 2, 1,"", text_halign=text.align_left, text_size=table_size(in_dashboardtab_size),


        table.cell(tab1, 0, 2, leadingindicator, text_halign=text.align_left, text_size=table_size(in_dash
        table.cell(tab1, 1, 2, leadingstatus, text_halign=text.align_left, text_size=table_size(in_dashboa
        table.cell(tab1, 2, 2, leadingstatus_short, text_halign=text.align_left, text_size=table_size(in_d


        table.cell(tab1, 0, 3, "Confirmation Indicators", text_halign=text.align_left, text_size=table_siz
        table.cell(tab1, 1, 3, "", text_halign=text.align_left, text_size=table_size(in_dashboardtab_size)
        table.cell(tab1, 2, 3, "", text_halign=text.align_left, text_size=table_size(in_dashboardtab_size)

        if array.size(confirmation_counter) > 0
            for i=0 to array.size(confirmation_counter)-1
                table.cell(tab1, 0, 4+i, array.get(confirmation_counter,i), text_halign=text.align_left, t
        else
            table.cell(tab1, 0, 4, "None Selected", text_halign=text.align_left, text_size=table_size(in_d


        if array.size(confirmation_val) > 0
            for j=0 to array.size(confirmation_val)-1
                table.cell(tab1, 1, 4+j, array.get(confirmation_val,j), text_halign=text.align_left, text_
                table.cell(tab1, 2, 4+j, array.get(confirmation_val_short,j), text_halign=text.align_left,


    plot(longCondition?100:na, "long Signal",display = display.data_window )
    plot(shortCondition?-100:na, "Short Signal",display = display.data_window )
```

PDF document made with CodePrint using Prism